Statistical Inference for Imputed Estimators Based on Regularized Imputation in Surveys

Zihe Wang

Thesis submitted to the University of Ottawa in partial fulfillment of the requirements for the degree of
Master of Science Mathematics and Statistics*

Department of Mathematics and Statistics
Faculty of Science
University of Ottawa

© Zihe Wang, Ottawa, Canada, 2026

*The M.Sc. program is a joint program with Carleton University, administered by the Ottawa-Carleton Institute of Mathematics and Statistics

Abstract

Penalized regression serves as a powerful alternative to ordinary least squares for handling multicollinearity. This property also holds true in survey sampling settings, where penalized regression is implemented for data imputation. In this paper, we analyze the behaviour of penalized regression estimators in survey contexts, focusing primarily on ridge regression and its debiased versions based on minimizing the conditional mean squared error (CMSE). Our study demonstrates that the proposed debiased approach performs well in controlling bias across various correlation structures and covariate dimensions, although it does not strictly outperform standard penalty selection methods such as cross-validation. Furthermore, we evaluate two methods for estimating the MSE of the imputed estimator: one based on Särndal's variance estimation framework and another relying on a pseudo-population bootstrap. The study of these MSE estimators illustrates how high-dimensional data impacts the validity and stability of these MSE estimators.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Survey sampling plays an important role in many areas such as public policy and economics, where the goal is to estimate characteristics of a finite population. In practice, however, survey data are often incomplete. One common issue is item nonresponse, where some sampled units do not answer certain questions. In some cases, this can affect the quality of the data and lead to biased estimates if not handled properly. (e.g., Little and Rubin [2019]).

A common way to deal with item nonresponse is to use imputation methods. In particular, linear deterministic imputation replaces missing values with predictions based on observed auxiliary variables. These predictions are usually obtained from linear regression models estimated by Ordinary Least Squares (OLS). While OLS works well in many situations, it can perform poorly when the auxiliary variables are highly correlated, which is often the case with modern, high-dimensional data.

When multicollinearity is present, the regression matrix becomes unstable, and the OLS estimator can have a large variance. This instability can negatively affect both the imputed values and the final population estimates. To address this issue, regularization methods such as Ridge regression and LASSO are often used. These methods introduce a penalty that reduces variance and stabilizes the estimates, at the cost of introducing some bias.Tibshirani [1996] Hoerl and Kennard [1970]

However, using regularized regression for imputation raises another issue. The bias introduced by regularization can propagate to the estimation of population totals, which is problematic in a survey sampling context. Therefore, it is important to develop methods that can handle multicollinearity while controlling this bias.

In this thesis, we propose an imputation method based on the m-th order debiased Ridge estimator. The goal is to reduce the bias introduced by regularization while keeping the stability benefits of Ridge regression. In addition, instead of selecting the tuning parameter using standard cross-validation, we introduce a Conditional Mean Squared Error (CMSE) criterion that is more appropriate for survey estimation. This criterion is developed under the mpq-framework, which accounts for the imputation model (m), the sampling design (p), and the nonresponse mechanism (q). It allows us to choose the tuning parameter in a way

that balances bias and variance for the estimator of a population mean.

The remainder of this thesis is organized as follows:

- Chapter 2 reviews the main concepts of linear and penalized regression in a classical setting, outside the survey sampling framework. In this chapter, we work with standard i.i.d. data and focus on the estimation of the regression coefficients $\boldsymbol{\beta}$, rather than on finite population parameters. We introduce the Ordinary Least Squares (OLS) estimator, the Ridge estimator, and the n-th order debiased Ridge estimator. A simulation study is then conducted to examine the behavior of the Ridge and debiased Ridge estimators under different levels of multicollinearity. The chapter concludes with a brief introduction to the LASSO and post-LASSO estimators, again in the context of estimating the regression coefficients $\boldsymbol{\beta}$.

- Chapter 3 introduces the basic concepts of survey sampling, including common sampling designs and classical estimators such as the Horvitz–Thompson estimator. The chapter also presents model-assisted estimators, which incorporate auxiliary information to improve efficiency. In particular, the Generalized Regression (GREG) estimator is introduced as a key example. Finally, we provide a brief review of the existing literature on model-assisted estimators based on regularization methods, including Ridge, debiased Ridge, and LASSO, as developed in the survey sampling context.

- Chapter 4 presents the first main methodological contribution of this thesis. In this chapter, we consider the problem of imputation for missing values in a survey sampling context. We study the properties of estimators obtained using Ridge regression and the m-th order debiased Ridge estimator. In particular, we propose a method for selecting the tuning parameter m based on the Conditional Mean Squared Error (CMSE), which allows us to determine an optimal value of m. A comprehensive simulation study is conducted to evaluate the performance of the proposed methodology. The results are compared with alternative approaches, including imputation based on LASSO and post-LASSO.

- Chapter 5 presents the second main contribution of this thesis. In practice, statistical agencies are interested not only in point estimation, but also in providing measures of uncertainty, such as variance or mean squared error (MSE) estimates. In this context, the use of regularized estimators such as Ridge or debiased Ridge may introduce bias, which must be properly accounted for when estimating the MSE. To address this issue, we develop two approaches for MSE estimation under the mpq-framework. The first approach is based on the variance decomposition proposed by Sarndal [1992]. The second approach relies on a resampling method, namely a pseudo-population bootstrap.

- Chapter 6 concludes the thesis and discusses possible directions for future work.

Chapter 2

Penalized Regression in Classical Statistics

2.1 Linear Regression: The Setup

In statistics, regression analysis is a technique used to examine the relationship between a response variable, denoted as y, and one or more explanatory variables x. The letter is also called predictors. The objectives of regression analysis are to estimate the value of the response variable, understand the nature of the relationships between the variables, conduct hypothesis testing to draw inferences about these relationships, or to predict future or unobserved values of the response variable based on the explanatory variables.

The ordinary linear regression model assumes that the relationship between the response variable y and the set of explanatory variables is linear. With n observations and p predictors, $x_1, x_2, \ldots, x_p$, the model is formulated as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}, \tag{2.1.1}$$

where

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \quad \mathbf{X} = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1p} \\ 1 & x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix},$$

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{bmatrix}, \quad \boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix}.$$

We make the following customary assumptions about the errors ε.

(i) $\mathbb{E}(\boldsymbol{\varepsilon} \mid \mathbf{X}) = \mathbf{0}$;

(ii) $\mathbb{V}(\boldsymbol{\varepsilon} \mid \mathbf{X}) = \mathbb{E}(\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^\top \mid \mathbf{X}) = \sigma^2 \mathbf{I}_n$, where $\mathbf{I}_n$ denotes the identity matrix of order n.

That is, we assume that the errors ε_i are uncorrelated and identically distributed. In classical regression, the design matrix is often treated as fixed, so we drop the conditioning on $\mathbf{X}$ when evaluating expectations. It follows from (2.1.1) that

$$\mathbb{E}(\mathbf{y}) = \mathbf{X}\boldsymbol{\beta},$$

$$\mathbb{V}(\mathbf{y}) = \mathbb{V}(\mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}) = \sigma^2 \mathbf{I}_n.$$

Often, the random errors $\varepsilon_i, i = 1, \ldots, n$, are assumed to follow a normal distribution, i.e., $\varepsilon_i \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma^2)$. While this assumption is useful for deriving exact finite-sample inference, it is not required for the unbiasedness and consistency of the ordinary least squares (OLS) estimator. When the sample size is sufficiently large, the normality assumption can be relaxed. In this case, the asymptotic normality of the OLS estimator, justified by the Central Limit Theorem, allows us to perform approximate statistical inference, such as constructing confidence intervals and conducting hypothesis tests.

Since the true values $\beta_j, j = 1, \ldots, p$, are unknown, they must be estimated from the observed data. We seek an estimator $\widehat{\boldsymbol{\beta}}$ of $\boldsymbol{\beta}$ that minimizes the residual sum of squares

$$\begin{aligned} S(\boldsymbol{\beta}) &= \sum_{i=1}^{n} \varepsilon_i^2 \\ &= \boldsymbol{\varepsilon}^\top \boldsymbol{\varepsilon} \\ &= (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^\top (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) \\ &= \mathbf{y}^\top \mathbf{y} - 2\boldsymbol{\beta}^\top \mathbf{X}^\top \mathbf{y} + \boldsymbol{\beta}^\top \mathbf{X}^\top \mathbf{X}\boldsymbol{\beta}. \end{aligned}$$

Taking the derivative $\partial S(\boldsymbol{\beta})/\partial \boldsymbol{\beta}$ and setting it equal to zero yields the normal equations:

$$\mathbf{X}^\top \mathbf{X}\boldsymbol{\beta} = \mathbf{X}^\top \mathbf{y}. \tag{2.1.2}$$

Assuming that the matrix $\mathbf{X}^\top \mathbf{X}$ is invertible, we obtain the OLS estimator of $\boldsymbol{\beta}$:

$$\widehat{\boldsymbol{\beta}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}. \tag{2.1.3}$$

The OLS estimator exists only when $\mathbf{X}^\top \mathbf{X}$ is invertible. If this matrix is singular, the OLS estimator cannot be computed. Singularity arises in two main situations:

(i) Perfect multicollinearity: one or more columns of $\mathbf{X}$ can be expressed as exact linear combinations of other columns. In this case, the rank of $\mathbf{X}$, denoted r, is smaller than p, making $\mathbf{X}^\top \mathbf{X}$ non-invertible.

(ii) High-dimensional setting: when $p > n$, the rank of $\mathbf{X}^\top \mathbf{X}$ is at most n, which is smaller than p. Consequently, its determinant is zero and the matrix is non-invertible.

Even when $\mathbf{X}^\top\mathbf{X}$ is invertible, numerical issues may arise. In practice, the columns of $\mathbf{X}$ are often nearly linearly dependent, a situation referred to as multicollinearity. In the presence of multicollinearity, or when the ratio p/n is not small (that is, $p/n \to \kappa \in (0,1)$), the matrix $\mathbf{X}^\top\mathbf{X}$ becomes ill-conditioned. This means that some predictors are nearly linear combinations of others, producing very small eigenvalues of $\mathbf{X}^\top\mathbf{X}$. When several eigenvalues are close to zero, the inverse $(\mathbf{X}^\top\mathbf{X})^{-1}$ becomes unstable, leading to large variances in the OLS estimator $\widehat{\boldsymbol{\beta}}$.

As $p/n \to \kappa$, results from random matrix theory show that the eigenvalues of $(1/n)\mathbf{X}^\top\mathbf{X}$ spread over an interval

$$[a(\kappa), b(\kappa)], \qquad \text{where } a(\kappa) \to 0 \text{ as } \kappa \text{ increases.}$$

This behavior is formalized by the Marchenko–Pastur law (Marchenko and Pastur [1967]), which describes the asymptotic distribution of the eigenvalues of sample covariance matrices. According to this law, as κ increases, the smallest eigenvalue $a(\kappa)$ approaches zero, indicating that the design matrix becomes increasingly ill-conditioned and the OLS estimator unstable. See also Bai and Silverstein [2010] for an in-depth treatment.

A common way to address this instability is through regularization, which introduces a penalty term to stabilize estimation (for example, ridge regression or lasso). These approaches reduce variance at the cost of introducing a small bias, improving the estimator's overall mean squared error in ill-conditioned settings.

In this thesis, we restrict attention to the classical case where $p < n$ and $\mathbf{X}^\top\mathbf{X}$ is of full rank, so that the OLS estimator in (2.1.3) exists and is well defined.

The OLS estimator $\widehat{\boldsymbol{\beta}}$ is unbiased for $\boldsymbol{\beta}$ since

$$\begin{aligned}
\text{Bias}\left(\widehat{\boldsymbol{\beta}}\right) &= \mathbb{E}\left(\widehat{\boldsymbol{\beta}}\right) - \boldsymbol{\beta} \\
&= (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbb{E}(\mathbf{y}) - \boldsymbol{\beta} \\
&= (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{X}\boldsymbol{\beta} - \boldsymbol{\beta} \\
&= \boldsymbol{\beta} - \boldsymbol{\beta} \\
&= \mathbf{0}.
\end{aligned} \tag{2.1.4}$$

The variance of $\widehat{\boldsymbol{\beta}}$ is given by

$$\begin{aligned}
\mathbb{V}\left(\widehat{\boldsymbol{\beta}}\right) &= \mathbb{V}\left\{(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}\right\} \\
&= \mathbb{V}\left\{(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top(\mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon})\right\} \\
&= \mathbb{V}\left\{(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\boldsymbol{\epsilon}\right\} \\
&= (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbb{V}(\boldsymbol{\epsilon})\mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1} \\
&= (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top(\sigma^2\mathbf{I}_n)\mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1}
\end{aligned}$$

$$= \sigma^2(\mathbf{X}^\top\mathbf{X})^{-1}. \tag{2.1.5}$$

According to the Gauss-Markov theorem, the OLS estimator, $\widehat{\boldsymbol{\beta}}$ is the Best Linear Unbiased Estimator (BLUE), meaning that it has the smallest variance among all linear and unbiased estimators of the coefficients.

2.2 Multicollinearity: The Effects

As mentioned above, a design matrix $\mathbf{X}$ is said to be ill-conditioned when its smallest singular approaches 0. The degree of ill-conditioning is measured by the condition number, defined as

$$\kappa(\mathbf{X}) = \frac{\sigma_{\max}(\mathbf{X})}{\sigma_{\min}(\mathbf{X})},$$

where $\sigma_{\max}(\mathbf{X})$ and $\sigma_{\min}(\mathbf{X})$ denote the largest and smallest singular values of $\mathbf{X}$, respectively. A matrix is considered well-conditioned when $\kappa(\mathbf{X})$ is close to 1, indicating similar scales among the singular values. Although there is no strict cutoff, condition numbers between 10 and 30 are typically interpreted as indicating moderate to strong ill-conditioning, while values exceeding 30 are often regarded as severely ill-conditioned in practice.

To illustrate the effect of multicollinearity on the behavior of the OLS estimator, consider the standard linear regression model with centered covariates $\tilde{x}_1, \ldots, \tilde{x}_p$, each having mean zero:

$$\mathbf{y} = \tilde{\mathbf{X}}\boldsymbol{\beta} + \boldsymbol{\varepsilon}, \tag{2.2.1}$$

where $\tilde{\mathbf{X}}$ is the centered design matrix. Since $\tilde{\mathbf{X}}^\top\tilde{\mathbf{X}}$ is symmetric, we can write its spectral decomposition as

$$\tilde{\mathbf{X}}^\top\tilde{\mathbf{X}} = \mathbf{P}\mathbf{D}\mathbf{P}^\top,$$

where $\mathbf{P}$ is an orthogonal matrix satisfying $\mathbf{P}^\top\mathbf{P} = \mathbf{P}\mathbf{P}^\top = \mathbf{I}_p$, and $\mathbf{D} = \operatorname{diag}(\lambda_1, \ldots, \lambda_p)$ contains the eigenvalues of $\tilde{\mathbf{X}}^\top\tilde{\mathbf{X}}$. We can rewrite (2.2.1) as

$$\begin{aligned}\mathbf{y} &= \tilde{\mathbf{X}}\mathbf{P}\mathbf{P}^\top\boldsymbol{\beta} + \boldsymbol{\varepsilon} \\ &= \mathbf{T}\boldsymbol{\eta} + \boldsymbol{\varepsilon},\end{aligned} \tag{2.2.2}$$

where $\mathbf{T} = \tilde{\mathbf{X}}\mathbf{P}$ and $\boldsymbol{\eta} = \mathbf{P}^\top\boldsymbol{\beta}$.

Applying OLS to (2.2.2) yields an unbiased estimator of $\boldsymbol{\eta}$:

$$\widehat{\boldsymbol{\eta}} = (\mathbf{T}^\top\mathbf{T})^{-1}\mathbf{T}^\top\mathbf{y}. \tag{2.2.3}$$

Now,

$$\begin{aligned}\mathbf{T}^\top\mathbf{T} &= \mathbf{P}^\top\tilde{\mathbf{X}}^\top\tilde{\mathbf{X}}\mathbf{P} \\ &= \mathbf{P}^\top(\mathbf{P}\mathbf{D}\mathbf{P}^\top)\mathbf{P}\end{aligned}$$

$$= \mathbf{D},$$

so that $\mathbf{T}^\top \mathbf{T} = \mathbf{D} = \text{diag}(\lambda_1, \ldots, \lambda_p)$, where λ_j denotes the eigenvalue associated with the j-th transformed predictor. It follows from (2.2.3) that

$$\begin{aligned} \widehat{\boldsymbol{\eta}} &= \mathbf{D}^{-1}\mathbf{T}^\top \mathbf{y} \\ &= \begin{pmatrix} \dfrac{\mathbf{t}_1^\top \mathbf{y}}{\lambda_1} \\ \vdots \\ \dfrac{\mathbf{t}_p^\top \mathbf{y}}{\lambda_p} \end{pmatrix}, \end{aligned} \tag{2.2.4}$$

where $\mathbf{t}_j^\top$ is the j-th row of $\mathbf{T}$. Equation (2.2.4) shows that when an eigenvalue λ_j is small, the corresponding estimated coefficient $\widehat{\eta}_j$ can take on a very large value.

The variance of $\widehat{\boldsymbol{\eta}}$ is given by

$$\begin{aligned} \mathbb{V}(\widehat{\boldsymbol{\eta}}) &= \mathbb{V}(\mathbf{D}^{-1}\mathbf{T}^\top \mathbf{y}) \\ &= \mathbf{D}^{-1}\mathbf{T}^\top \mathbb{V}(\mathbf{y})\mathbf{T}\mathbf{D}^{-1} \\ &= \sigma^2 \mathbf{D}^{-1}\mathbf{T}^\top \mathbf{T}\mathbf{D}^{-1} \\ &= \sigma^2 \mathbf{D}^{-1}\mathbf{D}\mathbf{D}^{-1} \\ &= \sigma^2 \mathbf{D}^{-1}. \end{aligned} \tag{2.2.5}$$

Thus, the variance of each estimated coefficient $\widehat{\eta}_j$ is proportional to $1/\lambda_j$, becoming very large when λ_j is small. In the extreme case of perfect multicollinearity, at least one eigenvalue equals zero, causing the corresponding coefficient to diverge to infinity.

Finally, the Mahalanobis norm of $\widehat{\boldsymbol{\eta}}$ is

$$\|\widehat{\boldsymbol{\eta}}\|_2^2 = \widehat{\boldsymbol{\eta}}^\top \left\{\sigma^2 \mathbf{D}^{-1}\right\}^{-1} \widehat{\boldsymbol{\eta}} = \frac{\mathbf{y}^\top \mathbf{T}\mathbf{D}^{-1}\mathbf{T}^\top \mathbf{y}}{\sigma^2},$$

which tends to be large in the presence of small eigenvalues, since $\mathbf{D}^{-1}$ contains the inverse of the eigenvalues of the design matrix. Therefore, small eigenvalues lead to large entries in $\mathbf{D}^{-1}$, which inflate the quadratic form. This observation suggests that one way to mitigate the adverse effects of multicollinearity is to constrain or penalize the size of $\widehat{\boldsymbol{\eta}}$, thereby preventing it from becoming excessively large. This idea forms the foundation of the ridge regression and lasso methods introduced in Sections 2.3 and 2.4.

2.3 Ridge Regression and Some Variants

Ridge regression, first proposed by Hoerl and Kennard [1970], is an alternative to ordinary least squares (OLS) estimation. It has gained renewed popularity in recent years because it stabilizes coefficient estimates in high-dimensional or highly collinear settings. Moreover, its single tuning parameter makes it straightforward to implement and interpret. In this section, we present the customary ridge estimator as well as some debiased versions.

2.3.1 The Customary Ridge Estimator

Ridge regression is a modification of OLS where a penalty is added to shrink the regression coefficients. This helps reduce the variance of the estimator and makes it more stable, especially when the explanatory variables are highly correlated. Specifically, we seek the coefficient vector $\boldsymbol{\beta}$ that minimizes

$$\min_{\boldsymbol{\beta}} \left\{ \sum_{i=1}^{n} (y_i - \mathbf{x}_i^\top \boldsymbol{\beta})^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}, \tag{2.3.1}$$

where $\lambda \geq 0$ is a tuning (regularization) parameter controlling the strength of the penalty. When $\lambda = 0$, the ridge estimator coincides with the OLS solution. As λ increases, the L_2 penalty progressively shrinks the coefficients toward zero, trading bias for reduced variance and more stable predictions. In practice, λ is typically selected to optimize predictive performance, often through K-fold cross-validation.

Differentiating (2.3.1) with respect to $\boldsymbol{\beta}$ and setting the derivative to zero yields

$$(\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_p) \widehat{\boldsymbol{\beta}}_R(\lambda) = \mathbf{X}^\top \mathbf{y}.$$

Hence, the ridge estimator is

$$\widehat{\boldsymbol{\beta}}_R(\lambda) = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_p)^{-1} \mathbf{X}^\top \mathbf{y}. \tag{2.3.2}$$

Unlike the OLS estimator, $\widehat{\boldsymbol{\beta}}_R(\lambda)$ is biased. Its bias is

$$\begin{aligned} \text{Bias}\{\widehat{\boldsymbol{\beta}}_R(\lambda)\} &= \mathbb{E}\{\widehat{\boldsymbol{\beta}}_R(\lambda)\} - \boldsymbol{\beta} \\ &= -\lambda (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_p)^{-1} \boldsymbol{\beta}. \end{aligned} \tag{2.3.3}$$

The variance of $\widehat{\boldsymbol{\beta}}_R(\lambda)$ is

$$\mathbb{V}\{\widehat{\boldsymbol{\beta}}_R(\lambda)\} = \sigma^2 (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_p)^{-1} \mathbf{X}^\top \mathbf{X} \big[(\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_p)^{-1} \big]^\top. \tag{2.3.4}$$

To better understand how the bias and variance of the ridge estimator behave, we consider two simple setups. Although these setups are not very realistic, they help illustrate the main ideas.

(1) *Orthonormal design*: Assume the columns of $\mathbf{X}$ are orthonormal, so that $\mathbf{X}^\top \mathbf{X} = \mathbf{I}_p$. Then

$$\text{Bias}\{\widehat{\boldsymbol{\beta}}_R(\lambda)\} = -\frac{\lambda}{1+\lambda} \boldsymbol{\beta}, \tag{2.3.5}$$

and

$$\mathbb{V}\{\widehat{\boldsymbol{\beta}}_R(\lambda)\} = \frac{\sigma^2}{(1+\lambda)^2} \mathbf{I}_p = \frac{1}{(1+\lambda)^2} \mathbb{V}(\widehat{\boldsymbol{\beta}}). \tag{2.3.6}$$

Thus, the variance of the ridge estimator is always smaller than that of OLS and goes to 0 as $\lambda \to \infty$, but this comes at the cost of a larger bias.

(2) *Standardized uncorrelated design*: Suppose each predictor is centered and scaled to have unit variance, so that $\sum_{i=1}^{n} x_{ij}^2 = n$, and the predictors are uncorrelated, leading to $\mathbf{X}^\top \mathbf{X} = n\mathbf{I}_p$. Then

$$\text{Bias}\{\widehat{\boldsymbol{\beta}}_R(\lambda)\} = -\frac{\lambda}{n+\lambda}\boldsymbol{\beta}, \tag{2.3.7}$$

$$\mathbb{V}\{\widehat{\boldsymbol{\beta}}_R(\lambda)\} = \frac{n}{(n+\lambda)^2}\sigma^2\mathbf{I}_p. \tag{2.3.8}$$

For fixed λ, both bias and variance vanish at rate $O(1/n)$ as $n \to \infty$. In this simplified setting, where $\mathbf{X}^\top\mathbf{X} = n\mathbf{I}_p$, both the bias and the variance of the ridge estimator decrease at rate $O(1/n)$ when λ is fixed. In particular, the estimator is consistent for any fixed value of λ.

However, consistency alone is not sufficient to guide the choice of the tuning parameter. In practice, the value of λ controls the trade-off between bias and variance, and its choice has a significant impact on the mean squared error (MSE) in finite samples. As the sample size increases, the variance of the estimator naturally decreases, and the need for regularization becomes less important. For this reason, it is desirable to allow the penalty parameter to decrease with n. This leads to considering $\lambda = \lambda_n$ such that

$$\frac{\lambda_n}{n} \to 0 \quad \text{as } n \to \infty.$$

In practice, the tuning parameter λ is typically selected by K-fold cross-validation (CV):

(i) Randomly split the n observations $(y_i, \mathbf{x}_i)$ into K approximately equal folds $\mathcal{F}_1, \ldots, \mathcal{F}_K$.

(ii) Choose a grid of Q penalty values $\{\lambda^{(1)}, \ldots, \lambda^{(Q)}\}$.

(iii) For each $\lambda^{(q)}$ and each fold k:

  (a) Fit ridge regression on the training data excluding fold k.

  (b) Compute the mean squared prediction error (MSE) on the held-out fold:

$$\text{MSE}_k^{(q)} = \frac{1}{|\mathcal{F}_k|}\sum_{i\in\mathcal{F}_k}\left(y_i - \mathbf{x}_i^\top \widehat{\boldsymbol{\beta}}_R^{(q,-k)}\right)^2.$$

(iv) Compute the average CV error:

$$\text{CV}(\lambda^{(q)}) = \frac{1}{K}\sum_{k=1}^{K}\text{MSE}_k^{(q)}.$$

(v) Select the value of λ that minimizes the CV error:

$$\widehat{\lambda} = \arg\min_{\lambda^{(q)}} \text{CV}(\lambda^{(q)}).$$

2.3.2 A First-Order Debiased Ridge Estimator

As discussed in Section 2.3.1, the ridge estimator given by

$$\widehat{\boldsymbol{\beta}}_R(\lambda) = (\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}_p)^{-1}\mathbf{X}^\top\mathbf{y}$$

is biased. Zhang and Politis [2022] proposed a **debiased ridge estimator** defined as

$$\widehat{\boldsymbol{\beta}}_{deb}(\lambda) = \widehat{\boldsymbol{\beta}}_R(\lambda) + \lambda(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}_p)^{-1}\widehat{\boldsymbol{\beta}}_R(\lambda). \tag{2.3.9}$$

Intuitively, $\widehat{\boldsymbol{\beta}}_{deb}(\lambda)$ can be viewed as taking the ridge estimator and "subtracting" an estimate of its first-order bias given by (2.3.3). An equivalent expression is

$$\widehat{\boldsymbol{\beta}}_{deb}(\lambda) = \left[\mathbf{I}_p + \lambda(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}_p)^{-1}\right](\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}_p)^{-1}\mathbf{X}^\top\mathbf{y}. \tag{2.3.10}$$

The debiased ridge estimator is still biased, but its bias is smaller. We have

$$\begin{aligned}\text{Bias}\{\widehat{\boldsymbol{\beta}}_{deb}(\lambda)\} &= \mathbb{E}\{\widehat{\boldsymbol{\beta}}_{deb}(\lambda)\} - \boldsymbol{\beta} \\ &= -\lambda^2(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}_p)^{-2}\boldsymbol{\beta}.\end{aligned} \tag{2.3.11}$$

Hence, the debiased estimator removes the first-order bias term of ridge regression and leaves only a second-order bias.

The variance of $\widehat{\boldsymbol{\beta}}_{deb}(\lambda)$ is given by

$$\begin{aligned}\mathbb{V}\{\widehat{\boldsymbol{\beta}}_{deb}(\lambda)\} = \sigma^2 &\left\{(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}_p)^{-1} + \lambda(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}_p)^{-2}\right\}\mathbf{X}^\top\mathbf{X} \\ &\times \left\{(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}_p)^{-1} + \lambda(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}_p)^{-2}\right\}.\end{aligned} \tag{2.3.12}$$

To better understand the properties of $\widehat{\boldsymbol{\beta}}_{deb}(\lambda)$, we examine two special cases already considered in Section 2.3.1.

(1) *Orthonormal design*: When $\mathbf{X}^\top\mathbf{X} = \mathbf{I}_p$, Expression (2.3.11) simplifies to

$$\text{Bias}\{\widehat{\boldsymbol{\beta}}_{deb}(\lambda)\} = -\left(\frac{\lambda}{1+\lambda}\right)^2\boldsymbol{\beta}.$$

Since $\lambda \geq 0$, the magnitude of this bias is smaller than that of $\widehat{\boldsymbol{\beta}}_R(\lambda)$. The variance (2.3.12) reduces to

$$\mathbb{V}\{\widehat{\boldsymbol{\beta}}_{deb}(\lambda)\} = \frac{(1+2\lambda)^2}{(1+\lambda)^4}\sigma^2\mathbf{I}_p,$$

which decreases as λ increases.

(2) *Standardized uncorrelated design*: If $\mathbf{X}^\top\mathbf{X} = n\,\mathbf{I}_p$, then

$$\text{Bias}\{\widehat{\boldsymbol{\beta}}_{deb}(\lambda)\} = -\frac{\lambda^2}{(n+\lambda)^2}\boldsymbol{\beta}. \tag{2.3.13}$$

For fixed λ, the bias of each component decreases at rate $O(1/n^2)$:

$$\text{Bias}\{\widehat{\beta}_{j,deb}(\lambda)\} = O(1/n^2), \qquad j = 1, \ldots, p.$$

The variance in this case is

$$\mathbb{V}\{\widehat{\boldsymbol{\beta}}_{deb}(\lambda)\} = \frac{n(n+2\lambda)^2}{(n+\lambda)^4}\sigma^2\mathbf{I}_p. \tag{2.3.14}$$

Comparing with the ridge variance $\mathbb{V}\{\widehat{\boldsymbol{\beta}}_R(\lambda)\} = \frac{n}{(n+\lambda)^2}\sigma^2\mathbf{I}_p$, we obtain

$$\frac{\mathbb{V}\{\widehat{\beta}_{j,deb}(\lambda)\}}{\mathbb{V}\{\widehat{\beta}_{j,R}(\lambda)\}} = \frac{(n+2\lambda)^2}{(n+\lambda)^2} \longrightarrow 1 \quad \text{as } n \to \infty.$$

Thus, as n increases, the variance difference between the two estimators becomes negligible, while the debiased ridge estimator has a much smaller bias.

2.3.3 An mth-Order Debiased Ridge Estimator

In Section 2.3.2, we introduced the first-order debiased ridge estimator $\widehat{\boldsymbol{\beta}}_{deb}(\lambda)$. Although its bias is smaller than that of the ridge estimator $\widehat{\boldsymbol{\beta}}_R(\lambda)$, it still has a remaining bias. This suggests extending the idea further by considering higher-order debiased ridge estimators.

Following Gao and Tsay [2024], we therefore consider higher-order bias corrections obtained by recursively correcting the remaining bias. For example, a second-order debiased ridge estimator is obtained by subtracting an estimate of the first-order bias from $\widehat{\boldsymbol{\beta}}_{deb}(\lambda)$. In the same way, one can define a third-order version, and more generally an mth-order debiased ridge estimator. The ordinary ridge estimator corresponds to the case $m = 0$, while the first-order debiased ridge estimator is obtained when $m = 1$. The general form is

$$\begin{aligned}\widehat{\boldsymbol{\beta}}_{m,deb}(\lambda) &= \widehat{\boldsymbol{\beta}}_R(\lambda) + \sum_{j=1}^{m} \lambda^j (\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}_p)^{-j}\widehat{\boldsymbol{\beta}}_R(\lambda) \\ &= \sum_{j=0}^{m} \lambda^j (\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}_p)^{-(j+1)}\mathbf{X}^\top\mathbf{y},\end{aligned} \tag{2.3.15}$$

where $\widehat{\boldsymbol{\beta}}_R(\lambda)$ is given by (2.3.2). Note that $\widehat{\boldsymbol{\beta}}_R(\lambda) = \widehat{\boldsymbol{\beta}}_{0,deb}(\lambda)$ and $\widehat{\boldsymbol{\beta}}_{deb}(\lambda) = \widehat{\boldsymbol{\beta}}_{1,deb}(\lambda)$.

The bias of $\widehat{\boldsymbol{\beta}}_{m,deb}(\lambda)$ is

$$\text{Bias}\{\widehat{\boldsymbol{\beta}}_{m,deb}(\lambda)\} = -\,\lambda^{m+1}(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}_p)^{-(m+1)}\boldsymbol{\beta}. \tag{2.3.16}$$

For $m = 0$ and $m = 1$, expression (2.3.16) reduces to (2.3.3) and (2.3.11), respectively. The variance of $\widehat{\boldsymbol{\beta}}_{m,deb}(\lambda)$ is given by

$$\mathbb{V}\{\widehat{\boldsymbol{\beta}}_{m,deb}(\lambda)\} = \sigma^2 \sum_{j=0}^{m} \sum_{k=0}^{m} \lambda^{j+k} (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_p)^{-(j+1)} \mathbf{X}^\top \mathbf{X} (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_p)^{-(k+1)}. \tag{2.3.17}$$

We conclude this section on the mth-order debiased ridge estimator with the following proposition, which shows that, as the number of bias-correction steps increases, the estimator converges to the OLS estimator, if $\mathbf{X}^\top \mathbf{X}$ is invertible.

Proposition 2.3.1. If $\mathbf{X}^\top \mathbf{X}$ is invertible, the m th–order debiased ridge estimator reproduces the OLS estimator in the limit as the order $m \to \infty$.

$$\widehat{\boldsymbol{\beta}}_{m,deb}(\lambda) \;=\; (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}, \text{ as } m \to \infty. \tag{2.3.18}$$

The proof is provided in Appendix A.1.

In the next section, we present a limited simulation study to compare the performance of the ridge estimator and several mth-order debiased versions in terms of bias and efficiency.

2.3.4 Empirical results

In this simulation experiment, we generated datasets with $n = 100$ observations and $p \in \{20, 40, 80\}$ covariates. Each row of the design matrix $\mathbf{X}$ was independently drawn from a multivariate normal distribution with mean vector $\mathbf{0}$, unit marginal variances, and a prespecified common pairwise correlation ρ.

To fix the values of the regression coefficients, the p covariates were first partitioned into two equal-sized groups. Values of the regression coefficients were then generated according to

$$\beta_j \;\sim\; \begin{cases} \text{Unif}(0,1), & j = 1, \ldots, \lfloor p/2 \rfloor, \\ \text{Unif}(1,2), & j = \lfloor p/2 \rfloor + 1, \ldots, p. \end{cases}$$

Given the values of $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)^\top$, the response was generated as

$$y_i = 10 + \mathbf{x}_i^\top \boldsymbol{\beta} + \varepsilon_i, \qquad \varepsilon_i \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0,1).$$

For each scenario, the simulation study was repeated over a $2 \times 3 \times 3$ grid of parameter settings:

(a) *Correlation level*: $\rho \in \{0.3, 0.95\}$;

(b) *Dimensionality ratio*: $p/n \in \{0.2, 0.4, 0.8\}$;

(c) *Ridge penalty*: $\lambda = 0.1n,\ 0.3n$, or λ chosen via five-fold cross-validation.

This led to 18 distinct simulation scenarios. The results are summarized in Figure 2.1, which reports the Monte Carlo MSE for all 18 simulation scenarios.

This data generation process was repeated independently $R = 2000$ times. For each simulated dataset, we computed the OLS estimator in (2.1.3) as well as the mth-order debiased ridge estimator in (2.3.15) for $m = 0, 1, \ldots, 50$.

Estimator performance was evaluated using the empirical mean squared error (MSE)

$$\mathrm{MSE}(\widehat{\boldsymbol{\beta}}) = \frac{1}{R}\sum_{r=1}^{R}\big\|\widehat{\boldsymbol{\beta}}^{(r)} - \boldsymbol{\beta}\big\|_2^2 = \frac{1}{R}\sum_{r=1}^{R}\sum_{j=1}^{p}\big(\widehat{\beta}_j^{(r)} - \beta_j\big)^2,$$

where $\widehat{\boldsymbol{\beta}}^{(r)}$ denotes the estimate from the rth replication, and $\widehat{\boldsymbol{\beta}}$ is used as a generic notation for the estimator under consideration.

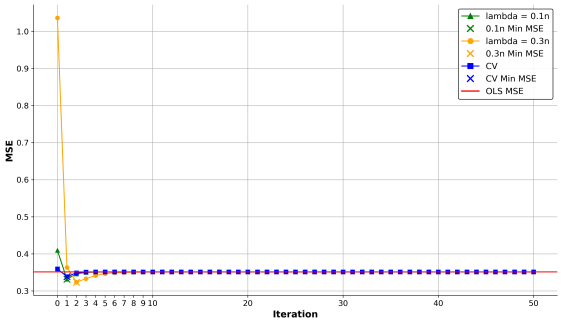
Figure 2.1 shows that the mth-order debiased ridge estimator can achieve a smaller mean squared error (MSE) than both the ridge and OLS estimators for appropriate values of m. In particular, for moderate values of m, the estimator often attains its lowest MSE, highlighting the benefit of bias correction. This behavior reflects the bias–variance trade-off inherent in the debiasing procedure. For small values of m, the estimator retains some of the bias reduction of ridge regression while avoiding the large variance of OLS. As m increases, the estimator approaches OLS, which may lead to an increase in MSE, especially in settings with strong multicollinearity. The gains in MSE are particularly noticeable when the correlation among covariates is moderate or high. In these cases, ridge regression substantially reduces variance, and the debiased ridge estimator further improves performance by partially correcting the bias. However, fully recovering the OLS estimator is not desirable, as OLS can exhibit very large variance. An exception appears in Figure 2.1(h), where the combination of high correlation ($\rho = 0.95$) and a large ratio $p/n = 0.8$ leads to a different behavior. In this case, the estimator based on cross-validation does not clearly benefit from increasing m, likely due to the extreme ill-conditioning of the design matrix.

2.4 LASSO Regression

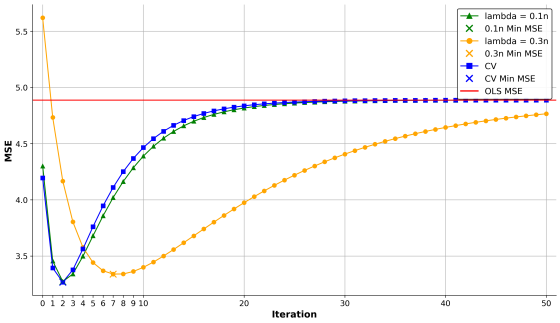
Another penalized regression method that can handle multicollinearity is the Least Absolute Shrinkage and Selection Operator (LASSO) proposed by Tibshirani [1996]. LASSO modifies the OLS objective function by adding an L_1 penalty on the regression coefficients:

$$\min_{\boldsymbol{\beta}}\left\{\sum_{i=1}^{n}(y_i - \mathbf{x}_i^\top\boldsymbol{\beta})^2 + \lambda\sum_{j=1}^{p}|\beta_j|\right\}, \tag{2.4.1}$$

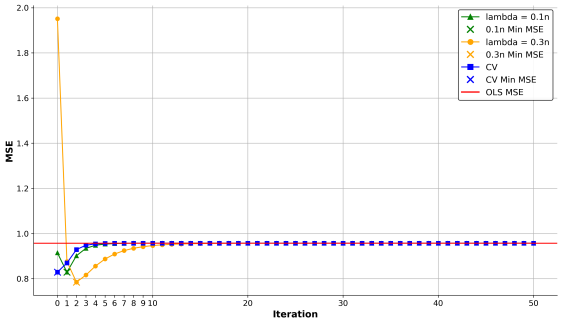
where $\lambda \geq 0$ controls the amount of regularization. As λ increases, some coefficients are shrunk exactly to zero, leading to a sparse model.



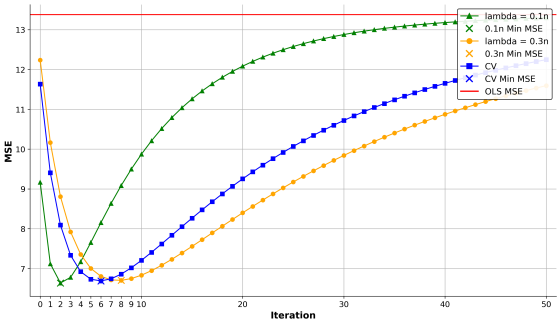
(a) Corr = 0.3 and $p/n = 0.2$, with OLS MSE = 0.35



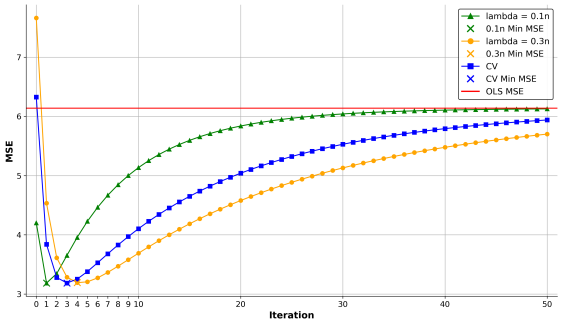
(b) Corr = 0.95 and $p/n = 0.2$, with OLS MSE = 4.9



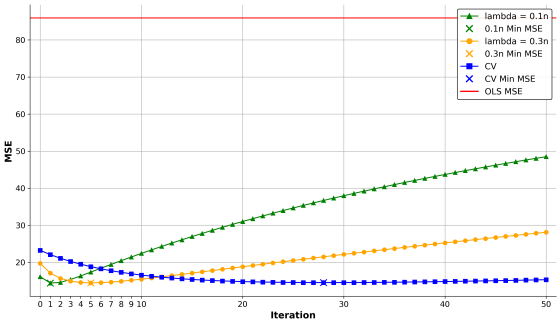
(c) Corr = 0.3 and $p/n = 0.4$, with OLS MSE = 0.95



(d) Corr = 0.95 and $p/n = 0.4$, with OLS MSE = 13.3



(e) Corr = 0.3 and $p/n = 0.8$, with OLS MSE = 6.1



(h) Corr = 0.95 and $p/n = 0.8$, with OLS MSE = 85.9

Figure 2.1: The MSE comparison for mth-order debiased ridge estimator to OLS across various correlation level, λ value, and covariates sizes.

Unlike ridge regression, the LASSO does not have a closed-form solution and must be computed numerically. A commonly used approach is coordinate descent, which updates one coefficient at a time while keeping the others fixed Friedman et al. [2010]. Each update involves a soft-thresholding step:

$$S(\beta_j, \lambda) = \text{sign}(\beta_j)\ \max(|\beta_j| - \lambda, 0).$$

This operation shrinks small coefficients toward zero and sets them exactly to zero when they fall below the threshold. Like ridge regression, the LASSO reduces variance by introducing bias. A key difference, however, is that it can set some coefficients exactly to zero, effectively removing variables from the model. This makes it useful for variable selection, especially when some predictors are not important.

When predictors are highly correlated, ridge regression tends to spread the effect across them, while the LASSO often selects only one and sets the others to zero. This behavior can be unstable, since small changes in the data may lead to different variables being selected. To address this issue, Zou and Hastie [2005] proposed the *Elastic Net*, which combines L_1 and L_2 penalties and tends to keep groups of correlated variables together.

The classical LASSO does not generally satisfy the so-called *oracle property*, meaning that it may fail to correctly identify the relevant variables and can shrink important coefficients too much. To address this limitation, Zou [2006] proposed the *Adaptive LASSO*, which uses data-dependent weights in the penalty:

$$\min_{\boldsymbol{\beta}} \left\{ \sum_{i=1}^{n} (y_i - \mathbf{x}_i^\top \boldsymbol{\beta})^2 + \lambda \sum_{j=1}^{p} w_j |\beta_j| \right\}, \tag{2.4.2}$$

where

$$w_j = \frac{1}{|\widehat{\beta}_{\text{OLS},j}|^\gamma}, \qquad \gamma > 0.$$

The idea is to penalize small coefficients more heavily and large coefficients less, based on an initial estimate. This reduces the bias on important predictors and improves variable selection.

Under suitable regularity conditions, the Adaptive LASSO enjoys the oracle property Zou [2006]. In particular:

(i) it identifies the correct set of nonzero coefficients with probability tending to one;

(ii) the estimated nonzero coefficients are asymptotically normal and as efficient as if the true model were known.

Although LASSO can perform variable selection, the shrinkage induced by the L_1 penalty introduces bias in the estimated coefficients. A simple way to reduce this bias is to run

a second-stage OLS regression using only the variables selected by LASSO. This approach, often referred to as the *post-LASSO OLS* estimator [Belloni and Chernozhukov, 2013, Belloni et al., 2014], proceeds as follows:

(i) Apply LASSO to the full set of predictors and identify the active set of selected variables

$$\widehat{S}_{\text{lasso}} = \{j : \widehat{\beta}_{\text{lasso},j} \neq 0\}.$$

(ii) Refit an unpenalized OLS regression restricted to the selected variables:

$$\arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} \left(y_i - \sum_{j \in \widehat{S}_{\text{lasso}}} x_{ij}\beta_j \right)^2 .$$

Chapter 3

Penalized Regression in Model-Assisted Estimation

In this chapter, we introduce the main elements of the design-based framework and the estimators that will be used throughout the thesis. We first define the finite population, the sampling design, and the inclusion probabilities. We then present the Horvitz–Thompson estimator, along with its variance and standard variance estimators. Next, we introduce the generalized regression (GREG) estimator, which uses auxiliary information to improve efficiency. Finally, we discuss situations where the GREG estimator can become unstable, especially when the covariates are highly correlated or when their number is large compared to the sample size. This motivates the use of penalized model-assisted estimators such as ridge and LASSO.

3.1 Survey Sampling: The Setup

Survey sampling is a branch of statistics concerned with making inference about a finite population. The population consists of a finite set of units, and the goal is to estimate quantities such as totals, means, or proportions. When nonsampling errors (e.g., measurement error or nonresponse) are absent, the standard approach is the design-based framework. In this setting, the population values are treated as fixed, and the only source of randomness comes from the sampling design used to select the sample. As a result, properties such as unbiasedness and variance are defined with respect to the sampling design, without relying on a superpopulation model.

Let $U = \{1, 2, ..., N\}$ denote a finite population composed of N units. Associated with each unit $i \in U$ is a value y_i, representing the measurement of a survey variable y. Within the design-based framework, these population y-values are treated as fixed. In this thesis, our objective is the estimation of the finite population total, $t_y = \sum_{i \in U} y_i$.

To estimate t_y, a probability sample $S \subset U$ of fixed size n is selected according to a specified

sampling design. We distinguish between S, viewed as a random variable representing the selected set of population units, and s, a particular realization of S. The collection of all samples that may be selected under the design forms the sample space

$$\Omega = \{\, s \subset U : p(s) > 0 \,\},$$

where $p(s)$ denotes the known probability of selecting sample s. By definition,

$$\sum_{s \in \Omega} p(s) = 1.$$

For each unit $i \in U$, define the sample selection indicator I_i by

$$I_i = \begin{cases} 1, & \text{if } i \in S, \\ 0, & \text{if } i \notin S. \end{cases}$$

The probability that unit i is included in the sample is called the first-order inclusion probability and is denoted by π_i :

$$\pi_i = P(i \in S) = \mathbb{E}(I_i) = \sum_{\substack{s \in \Omega \\ s \ni i}} p(s). \tag{3.1.1}$$

The sampling weight (or design weight) w_i associated with unit i is then defined as the reciprocal of its inclusion probability: $w_i = 1/\pi_i$.

Similarly, the second-order inclusion probability, π_{ij}, is defined as

$$\pi_{ij} = P(i \in S, j \in S) = \mathbb{E}(I_i I_j) = \sum_{\substack{s \in \Omega \\ s \ni i,j}} p(s). \tag{3.1.2}$$

In what follows, we denote the expectation, variance, and covariance with respect to the sampling design $p(\cdot)$ as $\mathbb{E}_p(\cdot), \mathbb{V}_p(\cdot), Cov_p(\cdot)$, respectively. We have

$$\begin{aligned} \mathbb{E}_p(I_i) &= \pi_i, \\ \mathbb{V}_p(I_i) &= \pi_i(1 - \pi_i), \\ \mathbb{E}_p(I_i I_j) &= \pi_{ij} \text{ for } i \neq j, \\ Cov_p(I_i, I_j) &= \pi_{ij} - \pi_i \pi_j \text{ for } i \neq j. \end{aligned} \tag{3.1.3}$$

A basic sampling design is simple random sampling without replacement (SRSWOR), where all samples of size n have the same probability of being selected. In this case, for any sample s,

$$p(S = s) = \frac{1}{\binom{N}{n}}. \tag{3.1.4}$$

The first- and second-order inclusion probabilities are given by

$$\begin{aligned} \pi_i &= \frac{\binom{N-1}{n-1}}{\binom{N}{n}} = \frac{n}{N} \text{ for all } i \in U, \\ \pi_{ij} &= \frac{\binom{N-2}{n-2}}{\binom{N}{n}} = \frac{n(n-1)}{N(N-1)} \text{ for all } i \neq j \in U. \end{aligned} \tag{3.1.5}$$

3.2 The Horvitz-Thompson estimator

A basic estimator of t_y, introduced by Horvitz and Thompson [1952], is the Horvitz–Thompson (HT) estimator given by

$$\widehat{t}_{y,HT} = \sum_{i \in S} \frac{y_i}{\pi_i} = \sum_{i \in S} w_i y_i. \tag{3.2.1}$$

Provided that $\pi_i > 0$ for all $i \in U$, the HT estimator is design-unbiased:

$$\mathbb{E}_p(\widehat{t}_{y,HT}) = \mathbb{E}_p\left(\sum_{i \in U} I_i \frac{y_i}{\pi_i}\right) = \sum_{i \in U} \mathbb{E}_p(I_i)\frac{y_i}{\pi_i} = \sum_{i \in U} y_i = t_y. \tag{3.2.2}$$

The variance of the Horvitz-Thompson estimator, $\widehat{t}_{y,HT}$, is derived by applying the general formula for the variance of a sum of (correlated) random variables, which leads to

$$\begin{aligned}
\mathbb{V}_p(\widehat{t}_{y,HT}) &= \mathbb{V}_p\left(\sum_{i \in U} I_i \frac{y_i}{\pi_i}\right) \\
&= \sum_{i \in U}\sum_{j \in U} \frac{y_i}{\pi_i}\frac{y_j}{\pi_j}\mathrm{Cov}_p(I_i, I_j) \\
&= \sum_{i \in U}\sum_{j \in U} (\pi_{ij} - \pi_i\pi_j)\frac{y_i}{\pi_i}\frac{y_j}{\pi_j}.
\end{aligned} \tag{3.2.3}$$

In the case of SRSWOR, Expression (3.2.3) reduces to

$$\mathbb{V}_p(\widehat{t}_{y,HT}) = N^2\left(1 - \frac{n}{N}\right)\frac{S_y^2}{n}, \tag{3.2.4}$$

where $(1 - n/N)$ is the finite population correction factor, and

$$S_y^2 = \frac{1}{N-1}\sum_{i \in U}(y_i - \bar{y}_U)^2$$

is the finite population variance for the survey variable y, with $\bar{y}_U = t_y/N$ denoting the population mean of y.

Provided that $\pi_{ij} > 0$ for all $i, j \in U$, an unbiased estimator of $\mathbb{V}_p(\widehat{t}_{y,HT})$ is

$$\widehat{\mathbb{V}}_p(\widehat{t}_{y,HT}) = \sum_{i \in S}\sum_{j \in S} \frac{\pi_{ij} - \pi_i\pi_j}{\pi_{ij}}\frac{y_i}{\pi_i}\frac{y_j}{\pi_j}. \tag{3.2.5}$$

To establish design-unbiasedness, we write the estimator in terms of the sample indicators and take expectations:

$$\mathbb{E}_p\left\{\widehat{\mathbb{V}}_p(\widehat{t}_{y,HT})\right\} = \sum_{i \in U}\sum_{j \in U} \frac{\pi_{ij} - \pi_i\pi_j}{\pi_{ij}}\frac{y_i}{\pi_i}\frac{y_j}{\pi_j} \tag{3.2.6}$$

$$= \mathbb{V}_p(\widehat{t}_{y,HT}). \tag{3.2.7}$$

In the case of SRSWOR, the variance estimator given by (3.2.5) reduces to the textbook variance estimator

$$\widehat{\mathbb{V}}_p(\widehat{t}_{y,HT}) = N^2 \left(1 - \frac{n}{N}\right) \frac{s_y^2}{n}, \tag{3.2.8}$$

where

$$s_y^2 = \frac{1}{n-1} \sum_{i \in S} (y_i - \bar{y}_s)^2$$

is the sample variance with $\bar{y}_s = \frac{1}{n} \sum_{i \in S} y_i$ denoting the sample mean.

3.3 The GREG estimator

In many survey applications, auxiliary variables are observed for the sampled units at the estimation stage, and the corresponding population totals are known from external sources. Such information can be incorporated into the construction of point estimators to improve efficiency. The generalized regression (GREG) estimator incorporates auxiliary information through a regression adjustment and typically has smaller variance than the Horvitz–Thompson estimator. More specifically, for each unit $i \in S$, a p-dimensional vector of auxiliary variables $\mathbf{x}_i = [x_{i1}, x_{i2}, \ldots, x_{ip}]^\top$ is observed, and the corresponding vector of population totals, $\mathbf{t}_\mathbf{x} = \sum_{i \in U} \mathbf{x}_i$, is assumed to be known.

Model-assisted estimation introduces a working superpopulation model, denoted $\mathcal{M}$, to describe the relationship between y and $\mathbf{x}$. For the GREG estimator, this model is taken to be the linear regression model

$$\mathcal{M}: \quad y_i = \mathbf{x}_i^\top \boldsymbol{\beta} + \epsilon_i, \tag{3.3.1}$$

where $\boldsymbol{\beta}$ is a p-vector of unknown regression coefficients and the model errors ϵ_i satisfy

(i) $\mathbb{E}_\mathcal{M}(\epsilon_i) = 0$,

(ii) $\mathbb{E}_\mathcal{M}(\epsilon_i \epsilon_j) = 0, \quad i \neq j$,

(iii) $\mathbb{V}_\mathcal{M}(\epsilon_i) = \sigma^2$,

with σ^2 an unknown variance parameter.

Model (3.3.1) is referred to as a superpopulation model. It is used as a working model to motivate the construction of the estimator. However, the properties of the estimator, such as bias and variance, are still evaluated with respect to the sampling design. In this sense, the model is only used to improve efficiency, without being required for the validity of the

inference.

If both $\mathbf{x}_i$ and y_i were known for all $i \in U$, one could fit an ordinary linear regression over the finite population. The resulting finite-population regression coefficient is denoted by $\tilde{\boldsymbol{\beta}}$. More specifically, let $\mathbf{X}_U$ be the $N \times p$ population matrix of auxiliary variables and let $\mathbf{y}_U$ be the $N \times 1$ vector of population y-values. The finite-population regression parameter $\tilde{\boldsymbol{\beta}}$ is the ordinary least squares estimator

$$\begin{aligned} \tilde{\boldsymbol{\beta}} &= \left(\mathbf{X}_U^\top \mathbf{X}_U\right)^{-1} \mathbf{X}_U^\top \mathbf{y}_U \\ &= \left(\sum_{i \in U} \mathbf{x}_i \mathbf{x}_i^\top\right)^{-1} \sum_{i \in U} \mathbf{x}_i y_i, \end{aligned} \tag{3.3.2}$$

Here, we assume that the $p \times p$ matrix $\left(\sum_{i \in U} \mathbf{x}_i \mathbf{x}_i^\top\right)$ is invertible.

In practice, the population parameter $\tilde{\boldsymbol{\beta}}$ in (3.3.2) cannot be computed, as the y-values are available for the sampled units $i \in S$ only. A natural estimator for $\tilde{\boldsymbol{\beta}}$, denoted $\widehat{\boldsymbol{\beta}}$, is obtained by replacing the two population totals in (3.3.2) with their respective Horvitz-Thompson estimators, leading to

$$\widehat{\boldsymbol{\beta}} = \left(\sum_{i \in S} w_i \mathbf{x}_i \mathbf{x}_i^\top\right)^{-1} \sum_{i \in S} w_i \mathbf{x}_i y_i. \tag{3.3.3}$$

In matrix notation, (3.3.3) can be written as

$$\widehat{\boldsymbol{\beta}} = \left(\mathbf{X}_S^\top \mathbf{W}_S \mathbf{X}_S\right)^{-1} \mathbf{X}_S^\top \mathbf{W}_S \mathbf{y}_S,$$

where $\mathbf{X}_S$ is the $n \times p$ matrix of sample auxiliary data, $\mathbf{y}_S$ is the $n \times 1$ sample vector of y_i values, and $\mathbf{W}_S = \mathrm{diag}(w_i)_{i \in S} = \mathrm{diag}(1/\pi_i)_{i \in S}$ is the $n \times n$ diagonal matrix of sampling weights.

The Generalized Regression (GREG) estimator of the population total t_y is given by

$$\widehat{t}_{\mathrm{greg}} = \sum_{i \in U} \mathbf{x}_i^\top \widehat{\boldsymbol{\beta}} + \sum_{i \in S} \frac{y_i - \mathbf{x}_i^\top \widehat{\boldsymbol{\beta}}}{\pi_i}. \tag{3.3.4}$$

The estimator (3.3.4) consists of two parts: (1) the sum of the model-predicted values, $\mathbf{x}_i^\top \widehat{\boldsymbol{\beta}}$, over the entire population, and (2) a Horvitz-Thompson correction term for the prediction residuals, $e_i = y_i - \widehat{y}_i$, observed in the sample. Expression (3.3.4) can be algebraically rearranged to write the GREG estimator as a weighted sum of the sample y-values:

$$\widehat{t}_{\mathrm{greg}} = \sum_{i \in S} w_i(S) y_i, \tag{3.3.5}$$

where

$$w_i(S) = w_i \left\{1 - \mathbf{x}_i^\top \left(\sum_{j \in S} w_j \mathbf{x}_j \mathbf{x}_j^\top\right)^{-1} \left(\sum_{j \in S} w_j \mathbf{x}_j - \sum_{j \in U} \mathbf{x}_j\right)\right\}, \quad i \in S.$$

Here, $w_i(S)$ denote the GREG weights, obtained by modifying the design weights $w_i = 1/\pi_i$. A third form for the GREG estimator is

$$\widehat{t}_{\text{greg}} = \widehat{t}_{y,HT} + \left(\sum_{j\in S} w_j \mathbf{x}_j - \sum_{j\in U} \mathbf{x}_j\right)^\top \widehat{\boldsymbol{\beta}}, \tag{3.3.6}$$

That is, the GREG estimator can be expressed as the Horvitz–Thompson estimator for t_y plus a regression adjustment based on the discrepancy between the known auxiliary totals $\sum_{j\in U} \mathbf{x}_j$ and their Horvitz–Thompson estimators $\sum_{j\in S} w_j \mathbf{x}_j$.

The GREG estimator is root-n consistent under suitable regularity conditions on the sampling design and the survey variables, even if the working model (3.3.1) is misspecified. More precisely, under a sequence of finite populations with population size $N \to \infty$ and sample size $n \to \infty$, the GREG estimator satisfies

$$\widehat{t}_{\text{greg}} - t_y = O_p(n^{-1/2}).$$

These results hold in a low-dimensional regime, where the number of auxiliary variables p is fixed as n increases. This property characterizes the model-assisted framework: the working model is used solely to construct an efficient estimator, while the validity of the resulting estimator is assessed with respect to the sampling design. In particular, consistency and asymptotic normality of the GREG estimator do not rely on correct specification of the working model; see, for example, Cassel et al. [1976], Robinson and Särndal [1983] and Breidt and Opsomer [2017].

It is convenient to work with the normalized estimator $N^{-1}\widehat{t}_{\text{greg}}$, which corresponds to the population mean. A first-order Taylor linearization leads to

$$\frac{1}{N}\widehat{t}_{\text{greg}} = \frac{1}{N}\widehat{t}_{y,HT} + \left(\frac{\sum_{j\in S} w_j \mathbf{x}_j - \sum_{j\in U} \mathbf{x}_j}{N}\right)^\top \tilde{\boldsymbol{\beta}} + O_p(n^{-1}), \tag{3.3.7}$$

Under standard regularity conditions, $\widehat{\boldsymbol{\beta}} - \tilde{\boldsymbol{\beta}} = O_p(n^{-1/2})$, and

$$N^{-1}\left(\sum_{j\in S} w_j \mathbf{x}_j - \sum_{j\in U} \mathbf{x}_j\right) = O_p(n^{-1/2}),$$

which implies an $O_p(n^{-1})$ remainder.

Dropping the remainder term in the linearization step leads to the usual linearized form of the GREG estimator:

$$\widehat{t}_{\text{greg}} \approx \mathbf{t}_{\mathbf{x}}^\top \tilde{\boldsymbol{\beta}} + \sum_{i\in S} \frac{E_i}{\pi_i}, \tag{3.3.8}$$

where

$$E_i = y_i - \mathbf{x}_i^\top \tilde{\boldsymbol{\beta}}, \qquad i \in U,$$

is the so-called census residual associated with unit i. Since $\mathbf{t}_x^\top \tilde{\boldsymbol{\beta}}$ is fixed under the sampling design, it does not contribute to the design variance. Therefore, the linearization approximation of the variance is given by

$$AV_p(\widehat{t}_{\text{greg}}) \approx \mathbb{V}_p\left(\sum_{i\in S}\frac{E_i}{\pi_i}\right) = \sum_{i\in U}\sum_{j\in U}(\pi_{ij}-\pi_i\pi_j)\,\frac{E_i}{\pi_i}\,\frac{E_j}{\pi_j}. \tag{3.3.9}$$

In the case of SRSWOR, Expression (3.3.9) reduces to

$$AV_p(\widehat{t}_{\text{greg}}) = N^2\left(1-\frac{n}{N}\right)\frac{S_E^2}{n}, \tag{3.3.10}$$

where

$$S_E^2 = \frac{1}{N-1}\sum_{i\in U}\left(E_i-\bar{E}_U\right)^2$$

with $\bar{E}_U = \frac{1}{N}\sum_{i\in U}E_i$.

The linearization variance expression in (3.3.9) involves the quantities $E_i = y_i - \mathbf{x}_i^\top\tilde{\boldsymbol{\beta}}$. In practice, these residuals are not available: $\tilde{\boldsymbol{\beta}}$ depends on the unknown population values $\{y_i : i \in U\}$, and therefore the E_i cannot be computed (neither for all $i \in U$ nor even for $i \in S$). Consequently, $AV_p(\widehat{t}_{\text{greg}})$ must be estimated. A standard approach is to replace E_i by the sample-based residuals

$$e_i = y_i - \mathbf{x}_i^\top\widehat{\boldsymbol{\beta}}, \qquad i\in S,$$

where $\widehat{\boldsymbol{\beta}}$ is the design-weighted regression coefficient computed from the sample. An estimator of $AV_p(\widehat{t}_{\text{greg}})$ is then

$$\widehat{AV}_p(\widehat{t}_{\text{greg}}) = \sum_{i\in S}\sum_{j\in S}\frac{(\pi_{ij}-\pi_i\pi_j)}{\pi_{ij}}\,\frac{e_i}{\pi_i}\,\frac{e_j}{\pi_j}. \tag{3.3.11}$$

For SRSWOR, (3.3.11) simplifies to

$$\widehat{AV}_p(\widehat{t}_{\text{greg}}) = N^2\left(1-\frac{n}{N}\right)\frac{s_e^2}{n}, \tag{3.3.12}$$

where

$$s_e^2 = \frac{1}{n-1}\sum_{i\in S}(e_i-\bar{e}_s)^2, \qquad \bar{e}_s = \frac{1}{n}\sum_{i\in S}e_i.$$

3.4 The Model-Assisted Estimator Based on Ridge Regression

The GREG estimator can become unstable in two situations. First, when the auxiliary variables exhibit strong multicollinearity, the design-weighted regression matrix $\mathbf{X}_S^\top\mathbf{W}_S\mathbf{X}_S$

becomes ill-conditioned, leading to an unstable estimator $\widehat{\boldsymbol{\beta}}$ even when the number of covariates p is small. Second, when the number of auxiliary variables p is large relative to the sample size n, the estimation error of $\widehat{\boldsymbol{\beta}}$ no longer vanishes sufficiently fast, introducing an additional source of variability in the GREG estimator. As a result, the variance reduction expected from the use of auxiliary information may be severely attenuated or even reversed.

To address this instability, penalized estimation methods can be adapted to the survey sampling framework. Analogous to the classical ridge solution (2.3.2), we define a finite-population ridge parameter, $\tilde{\boldsymbol{\beta}}_{ridge}(\lambda)$, as the target of estimation:

$$\begin{aligned}\tilde{\boldsymbol{\beta}}_{ridge}(\lambda) &= \left(\mathbf{X}_U^\top \mathbf{X}_U + \lambda \mathbf{I}_p\right)^{-1} \mathbf{X}_U^\top \mathbf{y}_U \\ &= \left(\sum_{i\in U} \mathbf{x}_i \mathbf{x}_i^\top + \lambda \mathbf{I}_p\right)^{-1} \sum_{i\in U} \mathbf{x}_i y_i, \end{aligned} \tag{3.4.1}$$

where $\lambda \geq 0$ is a tuning parameter controlling the strength of the penalty.

The ridge parameter $\tilde{\boldsymbol{\beta}}_{ridge}(\lambda)$ cannot be computed in practice, as it requires both the y-values and the auxiliary variables $\mathbf{x}_i$ to be available for all population units. As in the standard GREG framework, we therefore estimate each population quantity in (3.4.1) by its corresponding Horvitz–Thompson estimator, which leads to the design-weighted ridge estimator

$$\begin{aligned}\widehat{\boldsymbol{\beta}}_{ridge}(\lambda) &= \left(\mathbf{X}_S^\top \boldsymbol{\Pi}_S^{-1} \mathbf{X}_S + \lambda \mathbf{I}_p\right)^{-1} \mathbf{X}_S^\top \boldsymbol{\Pi}_S^{-1} \mathbf{y}_S \\ &= \left(\sum_{i\in S} w_i \mathbf{x}_i \mathbf{x}_i^\top + \lambda \mathbf{I}_p\right)^{-1} \sum_{i\in S} w_i \mathbf{x}_i y_i. \end{aligned} \tag{3.4.2}$$

Following the same structure as the standard GREG estimator, replacing the design-weighted least squares coefficient $\widehat{\boldsymbol{\beta}}$ in (3.3.6) by $\widehat{\boldsymbol{\beta}}_{ridge}(\lambda)$ yields the ridge-penalized GREG estimator

$$\widehat{t}_{\text{ridge}} = \sum_{i\in U} \mathbf{x}_i^\top \widehat{\boldsymbol{\beta}}_{ridge}(\lambda) + \sum_{i\in S} w_i \{y_i - \mathbf{x}_i^\top \widehat{\boldsymbol{\beta}}_{ridge}(\lambda)\}. \tag{3.4.3}$$

As for the GREG estimator, the ridge-penalized estimator (3.4.3) can be expressed as a weighted sum of the sample y-values:

$$\widehat{t}_{\text{ridge}} = \sum_{i\in S} w_i(S) y_i, \tag{3.4.4}$$

where

$$w_i(S) = \frac{1}{\pi_i} \left\{ 1 - \mathbf{x}_i^\top \left(\sum_{j\in S} w_j \mathbf{x}_j \mathbf{x}_j^\top + \lambda \mathbf{I}_p\right)^{-1} \left(\sum_{j\in S} w_j \mathbf{x}_j - \sum_{j\in U} \mathbf{x}_j\right) \right\}, \quad i \in S.$$

Finally, by rearranging the terms in (3.4.3), the ridge-penalized estimator can also be expressed as

$$\widehat{t}_{\text{ridge}} = \widehat{t}_{y,HT} + \left(\sum_{j\in S} w_j \mathbf{x}_j - \sum_{j\in U} \mathbf{x}_j\right)^{\top} \widehat{\boldsymbol{\beta}}_{ridge}(\lambda). \tag{3.4.5}$$

This expression clearly reveals that the estimator's structure is parallel to that of the GREG estimator (3.3.6).

If $\lambda = 0$, the estimator reduces to the standard GREG estimator, which is design-consistent but may exhibit high variance in the presence of multicollinearity. As $\lambda \to \infty$, the regression coefficients are increasingly shrunk toward zero and the ridge-penalized GREG estimator converges to the Horvitz–Thompson estimator. In this limit, no regression adjustment is used; the estimator remains design-unbiased but typically has larger variance than the GREG estimator when the auxiliary variables are effectively exploited. Therefore, the ridge-penalized GREG may be viewed as a trade-off between the GREG estimator and the Horvitz–Thompson estimator.

The performance of penalized model-assisted estimators depends critically on the choice of the tuning parameter λ. Several approaches have been proposed in the survey sampling literature for selecting λ in the ridge case, including weight-based diagnostic methods, numerical search procedures, and design-based criteria, see Bardsley and Chambers [1984], Beaumont and Bocci [2008], and Guggemos and Tillé [2010] , among others. Alternative algorithms have been developed to ensure stable calibration weights or to optimize design-based objective functions. For LASSO-type estimators (McConville et al. [2017]), cross-validation methods have been commonly used.

Dagdoug et al. [2023] study the design-based properties of the ridge-penalized GREG estimator, when the number of auxiliary variables grows with the sample size. Under mild moment conditions on the survey variable, standard regularity assumptions on the sampling design, and suitable growth conditions on the auxiliary variables, they show that the ridge-penalized GREG estimator is design-consistent in high-dimensional settings. More precisely,

$$N^{-1}\big(\widehat{t}_{\text{ridge}} - t_y\big) = O_p\left(\sqrt{\frac{p^3}{n}}\right),$$

which implies consistency whenever $p^3/n \to 0$. This result holds without requiring correct specification of the working regression model and highlights the stabilizing role of ridge regularization in the presence of multicollinearity or a large number of auxiliary variables.

Finally, variance estimation for the ridge-penalized GREG estimator can be carried out using standard design-based approximations. In practice, it is common to use the GREG variance estimator given in (3.3.11), replacing the OLS residuals with the ridge residuals $e_i = y_i - \mathbf{x}_i^{\top}\widehat{\boldsymbol{\beta}}_{\text{ridge}}(\lambda)$ Goga [2011]. This plug-in variance estimator ignores the additional variability due to the choice of λ, but it generally provides a reasonable approximation of the mean squared error when λ is small.

3.5 The LASSO Estimator

The LASSO provides an alternative penalized approach to model-assisted estimation in survey sampling. As with the ridge-based estimator discussed in Section 2.4, the survey-LASSO estimator is constructed by replacing the standard design-weighted least squares coefficient $\widehat{\boldsymbol{\beta}}$ in the GREG estimator with a penalized regression coefficient, with the goal of improving stability and efficiency in high-dimensional settings.

The design-weighted LASSO coefficient, denoted by $\widehat{\boldsymbol{\beta}}_{\text{Lasso}}(\lambda)$, is defined as the solution to

$$\widehat{\boldsymbol{\beta}}_{\text{Lasso}}(\lambda) = \arg\min_{\boldsymbol{\beta}} \left\{ \sum_{i \in S} w_i \left(y_i - \mathbf{x}_i^\top \boldsymbol{\beta}\right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\}, \tag{3.5.1}$$

where $\lambda \geq 0$ is a tuning parameter. Unlike the ridge estimator, the LASSO estimator does not possess a closed-form analytical solution.

Once the coefficient $\widehat{\boldsymbol{\beta}}_{Lasso}(\lambda)$ is obtained, the survey-LASSO estimator for the population total is constructed using the standard GREG formulation:

$$\widehat{t}_{\text{lasso}} = \widehat{t}_{y,HT} + \left(\sum_{j \in S} w_j \mathbf{x}_j - \sum_{j \in U} \mathbf{x}_j \right)^\top \widehat{\boldsymbol{\beta}}_{Lasso}(\lambda). \tag{3.5.2}$$

The theoretical properties of the survey-weighted LASSO estimator in a design-based framework were first established by McConville et al. [2017]. Motivated by the growing availability of high-dimensional auxiliary data—such as remote sensing information used in the U.S. Forest Service's Forest Inventory and Analysis program—they study model-assisted estimation of finite-population totals using a LASSO regression adjustment and compare it to the standard GREG estimator. Under an asymptotic framework with nested populations and a fixed number of auxiliary variables, they establish design consistency and asymptotic normality of the LASSO-based total estimator. While the LASSO estimator is asymptotically equivalent to the GREG estimator, it can substantially improve finite-sample performance by shrinking or eliminating irrelevant covariates, leading to more stable weights and potential efficiency gains. They also propose a design-consistent variance estimator for the LASSO estimator. A distinct challenge with the LASSO estimator is that, unlike the GREG or Ridge estimators, the solution $\widehat{\boldsymbol{\beta}}_{Lasso}$ is not a linear function of the response variable y. Consequently, the estimator cannot naturally be expressed as a weighted linear sum of the sample sample observations, which complicates the production of general-purpose survey weights. To address this, McConville et al. [2017] proposed alternative a model-calibration approach, which allows for the generation of a single set of weights applicable to multiple study variables. In the high-dimensional survey setting studied by Dagdoug et al. [2023], the model-assisted LASSO estimator is design-consistent provided that $p^3/n \to 0$, even when the number of auxiliary variables diverges with the sample size.

Although the LASSO estimator is attractive for its variable selection feature, it is known to induce bias in the estimation of coefficients with large magnitude due to the uniform L_1

penalty; see Section 2.4. To reduce this bias while preserving sparsity, the Adaptive LASSO (ALASSO) can be extended to the survey sampling context. In a design-based framework, the Adaptive LASSO modifies the penalty by introducing coefficient-specific weights. This approach is motivated by the oracle property Zou [2006], which states that, under suitable conditions, an estimator can asymptotically identify the correct set of nonzero coefficients and estimate them with the same efficiency as if that set were known in advance.

McConville et al. [2017] extended the survey-weighted LASSO to an adaptive LASSO framework by introducing data-driven weights in the L_1 penalty, designed to reduce shrinkage bias for large regression coefficients. Under standard design-based asymptotics with fixed p, they show that the adaptive LASSO GREG estimator is design-consistent and asymptotically normal for finite-population totals. As with the standard LASSO, shrinkage and variable selection are asymptotically negligible for totals, while finite-sample performance can improve through increased stability when the working model is sparse. The survey-weighted Adaptive LASSO coefficient, $\widehat{\boldsymbol{\beta}}_{ALasso}(\lambda)$, is obtained by solving:

$$\widehat{\boldsymbol{\beta}}_{ALasso}(\lambda) = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \left(\sum_{i \in S} w_i \left(y_i - \mathbf{x}_i^\top \boldsymbol{\beta} \right)^2 + \lambda \sum_{j=1}^{p} \widehat{w}_j |\beta_j| \right), \tag{3.5.3}$$

where the adaptive penalty weights are defined as $\widehat{w}_j = 1/|\tilde{\beta}_j|^\gamma$, with $\gamma > 0$ (commonly $\gamma = 1$) Zou [2006], McConville et al. [2017]. By assigning larger penalties to coefficients that are likely to be zero (where $\tilde{\beta}_j$ is small) and smaller penalties to non-zero coefficients, the estimator aims to reduce the estimation bias.

Once the coefficient vector is computed, the Adaptive LASSO estimator for the population total is constructed using the GREG formulation:

$$\widehat{t}_{\text{alasso}} = \widehat{t}_{y,HT} + \left(\sum_{j \in S} w_j \mathbf{x}_j - \sum_{j \in U} \mathbf{x}_j \right)^\top \widehat{\boldsymbol{\beta}}_{ALasso}(\lambda). \tag{3.5.4}$$

McConville et al. [2017] proposed this estimator and evaluated it via simulation, suggesting it behaves similarly to the standard LASSO.

A few other related contributions have appeared in the literature. For instance, Chen et al. [2018] extended the adaptive LASSO approach to model-assisted calibration for non-probability samples. More recently, Lundy and Rao [2022] reported simulation results comparing LASSO-based estimators with the standard GREG estimator in high-dimensional settings.

Chapter 4

Imputation Based on Regularized Regression

4.1 Treatment of Missing Data

In survey sampling, missing data arise when respondents fail to complete the survey in full or when specific answers are not recorded. Such missing values can lead to invalid statistical inferences, as unadjusted estimates may suffer from significant bias. Missing data in survey sampling can be categorized into two types: unit nonresponse and item nonresponse. Table (4.1) presents a typical dataset structure following survey data collection. The dataset consists of n sampled units and p survey variables, $y_1, y_2, ..., y_p$.

	y_1	y_2	y_3	$\cdots$	y_p	
1	✓	✓	✓	...	✓	{ Full response }
2	✓	✓	✓	...	✓	
⋮	✓	×	×	...	✓	{ Item nonresponse }
⋮	×	✓	✓	...	×	
⋮	×	×	×	...	×	{ Unit nonresponse }
n	×	×	×	...	×	

Table 4.1: Levels of nonresponse

Unit nonresponse occurs when a sampled unit (e.g., the i -th observation) fails to participate in the survey. This may result from refusal to respond or inability to contact the unit. A common approach to addressing unit nonresponse involves weight adjustment procedures, which consists of inflating the weights of responding units to account for the units that did not respond. That is, the sampling weights $w_i = 1/\pi_i$ are adjusted either through a calibration procedure or through response propensity modeling; see,e.g., Haziza and Beaumont

(2017, Statistical Science) for an overview.

Item nonresponse occurs when a respondent participates in the survey but fails to provide answers to specific questions. This type of missingness may arise due to sensitive questions or recall difficulties. A common approach to addressing item nonresponse is single imputation, where each missing response is replaced by a plausible value derived from the observed data. In practice, an imputation model is first fitted to the respondents' records, using fully observed auxiliary variables to predict the values of non-respondents. Auxiliary variables are variables that are recorded for all the sample units (respondents or and nonrespondents).

In this chapter, we focus on the treatment of item nonresponse through imputation. Consider a variable of interest y that is subject to missingness. Let r_i denote the response indicator for unit i where $r_i = 1$ if y_i is observed, and $r_i = 0$ if y_i is missing. Let $S_r \subseteq S$ denote the set of responding units, of size n_r, for which y_i is observed (i.e., those with $r_i = 1$), and let $S_m \subseteq S$ denote the set of nonresponding units, of size n_m, for which y_i is missing (i.e., those with $r_i = 0$). We have $n = n_r + n_m$.

We define the response probability for each unit i as

$$p_i = P(r_i = 1|y_i, \mathbf{x}_i),$$

where $\mathbf{x}_i$ is a vector of fully observed variables for unit i, i.e., variables that are observed for both respondents and nonrespondents. Rubin [1976] classified the missing mechanisms into one of three categories: Missing Completely at Random (MCAR), Missing at Random (MAR), and Missing Not at Random (MNAR).

Missing Completely at Random (MCAR) refers to a situation in which the probability of an observation being missing is independent of both the observed and unobserved data. Formally, this condition can be expressed as

$$p_i = P(r_i = 1|y_i, \mathbf{x}_i) = P(r_i = 1). \tag{4.1.1}$$

Conceptually, an MCAR mechanism can be viewed as the random selection of a subsample from the observed sample. A typical example of MCAR arises when the response probabilities satisfy $p_i = p_0$ for all $i \in U$. Under MCAR, missingness does not introduce bias into statistical estimators. Consequently, complete-case analysis remains valid, although it may suffer from a loss of efficiency due to the reduction in effective sample size.

Missing at Random (MAR) refers to a missing data mechanism in which the probability of missingness depends only on fully observed variables, but not on the survey variable y that is prone to missingness. Formally, this condition can be expressed as Rubin [1976]:

$$p_i = P(r_i = 1|y_i, \mathbf{x}_i) = P(r_i = 1|\mathbf{x}_i). \tag{4.1.2}$$

This condition implies that, after conditioning on the fully observed variables $\mathbf{x}_i$, the missingness mechanism no longer depends on the unobserved value y_i. In contrast to MCAR, the MAR assumption allows the probability of missingness to vary across subgroups defined by $\mathbf{x}_i$. A common example is income nonresponse in surveys, where the probability of responding depends on education level, which is fully observed. Individuals with higher levels of education may be more likely to report their income. However, conditional on education, the probability of nonresponse does not depend on the true income value itself, so that missingness is at random within education groups. Under MAR, valid statistical inference is still possible, provided that the imputation model includes the relevant variables $\mathbf{x}_i$ and is correctly specified.

Missing Not at Random (MNAR) arises when the probability of missingness depends directly on the unobserved values themselves, even after conditioning on $\mathbf{x}$. Formally, this mechanism is characterized by

$$p_i = P(r_i = 1|y_i, \mathbf{x}_i) \neq P(r_i = 1|\mathbf{x}_i). \tag{4.1.3}$$

In contrast to MCAR and MAR, under the Missing Not at Random (MNAR) mechanism, the probability of missingness is systematically related to the unobserved values themselves. As a result, standard statistical adjustments that rely solely on the observed data are generally insufficient unless the missing data process is explicitly modeled. A typical example arises when individuals with higher incomes are less likely to report their earnings, even after accounting for education and other demographic characteristics. Because MNAR mechanisms can introduce substantial bias, addressing them typically requires strong model-based assumptions and/or the use of external data sources Rubin [1976].

4.2 Imputation Procedures

When values of the survey variable y are missing, an estimator of the population total, $t_y = \sum_{i \in U} y_i$, is given by

$$\widehat{t}_I = \sum_{i \in S} w_i r_i y_i + \sum_{i \in S} w_i (1 - r_i) y_i^* = \sum_{i \in S} w_i \tilde{y}_i, \tag{4.2.1}$$

where $\tilde{y}_i = r_i y_i + (1 - r_i) y_i^*$ with y_i^* denoting the imputed value for the missing y_i. The estimator $\widehat{t}_I$ in (4.2.1) is called an imputed estimator.

A wide variety of imputation methods have been proposed in the literature to construct the imputed values y_i^* for the missing observations. Imputation methods are generally classified into two broad categories: deterministic and random procedures. Under deterministic imputation, repeated application of the same method to the same dataset produces identical imputed values and corresponding estimates. In contrast, random imputation introduces stochastic component into the imputation process, leading to different completed datasets

across repeated runs on the same data.

Alternatively, imputation methods can also be classified as either donor-based or model-based (predicted value) approaches. Donor imputation replaces a missing value for a recipient(a nonrespondent) unit with an observed value from a similar donor(respondent) unit, ensuring that all imputed values originate from actual responses within the dataset. In contrast, predicted value imputation relies on a statistical model to estimate missing values, using relationships inferred from the observed data rather than directly substituting observed responses.

In this chapter, we focus on model-based deterministic parametric imputation under the MAR assumption. Specifically, we assume that the relationship between the survey variable y requiring imputation and the vector of fully observed variables may be described as

$$y_i = m(\mathbf{x}_i, \boldsymbol{\beta}) + \epsilon_i, \tag{4.2.2}$$

where $m(\mathbf{x}_i, \boldsymbol{\beta})$ is a prespecified function with $\boldsymbol{\beta}$ denoting the vector of regression coefficients. The error term ϵ_i in (4.2.2) satisfies the following conditions:

$$\mathbb{E}(\epsilon_i|\mathbf{x}_i) = 0, \quad \mathbb{E}(\epsilon_i\epsilon_j|\mathbf{x}_i\mathbf{x}_j) = 0, \text{ for i} \neq j, \text{ and } \mathbb{V}(\epsilon_i|\mathbf{x}_i) = \sigma^2 c_i, \tag{4.2.3}$$

where $c_i > 0$ is a known coefficient associated with unit i. In this thesis, we assume that $c_i = 1$ for all i. Note that we do not make any distributional assumptions about the error terms ϵ_i. Deterministic parametric imputation involves replacing the missing values y_i with

$$y_i^* = m(\mathbf{x}_i, \widehat{\boldsymbol{\beta}}_r), i \in S_m, \tag{4.2.4}$$

where $\widehat{\boldsymbol{\beta}}_r$ is a suitable estimator of $\boldsymbol{\beta}$ (e.g., least square estimators, maximum likelihood estimators).

To establish the properties of $\widehat{t}_I$, we adopt the so called *mpq*-inferential framework (e.g.,Chen and Haziza [2019]), which accounts for the combined effects of the model (m), sampling design (p), and nonresponse mechanism (q). Under this framework, the bias of the imputed estimator is defined as

$$Bias(\widehat{t}_I) = \mathbb{E}_{mpq}(\widehat{t}_I - t_y) \equiv \mathbb{E}_m\mathbb{E}_p\mathbb{E}_q(\widehat{t}_I - t_y), \tag{4.2.5}$$

where $\mathbb{E}_m(\cdot)$, $\mathbb{E}_p(\cdot)$, and $\mathbb{E}_q(\cdot)$ denote expectations taken with respect to the imputation model, the sampling design, and the nonresponse mechanism, respectively. To clarify the nature of the expectations $\mathbb{E}_m(\cdot)$, $\mathbb{E}_p(\cdot)$, and $\mathbb{E}_q(\cdot)$, we first identify all the quantities involved in the imputed estimator $\widehat{t}_I$:

$$\mathbf{I} = (I_1, I_2, \ldots, I_N)^\top, \quad \mathbf{r} = (r_1, r_2, \ldots, r_N)^\top, \quad \mathbf{y} = (y_1, y_2, \ldots, y_N)^\top,$$

where $\mathbf{I}$ is the vector of sample-selection indicators ($I_i = 1$ if unit i is in the sample, 0 otherwise); $\mathbf{r}$ is the vector of response indicator ($r_i = 1$ when y_i is observed, 0 when it is missing); and $\mathbf{y}$ is the vector of the population y-values. In addition, we denote by $\mathbf{X}$ the matrix of auxiliary variables used at the imputation stage, and $\mathbf{Z}$ the matrix of auxiliary variables used at the design stage. The following table summarizes which quantities are treated as fixed or random under each type of expectation:

	I	**y**	**r**	**X**	**Z**
$\mathbb{E}_m(\cdot)$	Fixed	Random	Fixed	Fixed	Fixed
$\mathbb{E}_p(\cdot)$	Random	Fixed	Fixed	Fixed	Fixed
$\mathbb{E}_q(\cdot)$	Fixed	Fixed	Random	Fixed	Fixed

Table 4.2: Classification of variables as fixed or random under different expectation operators.

4.3 Linear Regression Imputation

In this section and subsequent sections, we study a special case of (4.2.4), namely deterministic linear regression imputation, whereby

$$m(\mathbf{x}_i, \widehat{\boldsymbol{\beta}}_r) = \mathbf{x}_i^\top \widehat{\boldsymbol{\beta}}_r \tag{4.3.1}$$

with

$$\widehat{\boldsymbol{\beta}}_r = \left(\sum_{i \in S_r} w_i \mathbf{x}_i c_i^{-1} \mathbf{x}_i^\top \right)^{-1} \sum_{i \in S_r} w_i \mathbf{x}_i c_i^{-1} y_i, \tag{4.3.2}$$

denoting the weighted least squares estimator of $\widehat{\boldsymbol{\beta}}$. This leads to the imputed values

$$y^*_{i,linear} = \mathbf{x}_i^\top \widehat{\boldsymbol{\beta}}_r, i \in S_m. \tag{4.3.3}$$

The resulting imputed estimator of t_y is given by

$$\widehat{t}_I^{lin} = \sum_{i \in S} w_i r_i y_i + \sum_{i \in S} w_i (1 - r_i) y^*_{i,linear}. \tag{4.3.4}$$

Proposition 4.3.1. Assume that the missing data mechanism is MAR and that the first moment of the imputation model is correctly specified, that is,

$$\mathbb{E}_m(y_i \mid \mathbf{x}_i) = \mathbf{x}_i^\top \boldsymbol{\beta}$$

for some vector $\boldsymbol{\beta}$. Then, under the mpq-inferential framework, $\widehat{t}_I^{lin}$ is mpq-unbiased for t_y:

$$\text{Bias}(\widehat{t}_I^{\text{lin}}) = \mathbb{E}_{mpq}\big(\widehat{t}_I^{\text{lin}} - t_y\big) = 0. \tag{4.3.5}$$

The proof is provided in Appendix A.2.

Although the linear regression imputation estimator is unbiased when the imputation model is correctly specified, it may still perform poorly in practice. In particular, as discussed in Section 2.2, severe multicollinearity among the covariates can lead to unstable estimates of the regression coefficients and, consequently, to inflated variance. This instability propagates to the imputed values and, in turn, to the imputed estimator, resulting in poor finite-sample performance. In such situations, penalized regression methods, such as Ridge or LASSO, provide a more robust alternative by stabilizing estimation through regularization and reducing variance.

4.4 Ridge and mth-Order Debiased Ridge Imputation

Ridge imputation extends the linear regression imputation framework by fitting a ridge-penalized regression model to the respondent sample. This parallels the use of ridge regression in classical linear models (Section 2.3) and in model-assisted estimation for survey sampling (Section 3.4), where regularization is employed to control variance inflation caused by near-singular covariate matrices.

In the imputation context, the usual least-squares estimator $\widehat{\boldsymbol{\beta}}_r$ is replaced by the ridge estimator

$$\widehat{\boldsymbol{\beta}}_r^R(\lambda) = \left(\sum_{i\in S_r} w_i \mathbf{x}_i \mathbf{x}_i^\top + \lambda \mathbf{I}_p\right)^{-1} \sum_{i\in S_r} w_i \mathbf{x}_i y_i, \tag{4.4.1}$$

where $\lambda > 0$ denotes a penalty parameter.

For each nonrespondent unit $i \in S_m$, the imputed values are given by

$$y_{i,Ridge}^* = \mathbf{x}_i^\top \widehat{\boldsymbol{\beta}}_r^R(\lambda). \tag{4.4.2}$$

The corresponding ridge imputed estimator of the population total t_y is

$$\widehat{t}_I^R = \sum_{i\in S} w_i r_i y_i + \sum_{i\in S} w_i (1-r_i) \mathbf{x}_i^\top \widehat{\boldsymbol{\beta}}_r^R(\lambda). \tag{4.4.3}$$

The penalty parameter λ is generally unknown and may depend on the realized sample and the set of respondents. In practice, λ is often selected using data-driven procedures such as cross-validation and is therefore sample-dependent. Since its distribution under the *mpq* framework is typically intractable, we treat λ as fixed when studying the bias of (4.4.3).

The total error of $\widehat{t}_I^R$ can be expressed as

$$\widehat{t}_I^R - t_y = (\widehat{t}_{y,\mathrm{HT}} - t_y) + (\widehat{t}_I^R - \widehat{t}_{y,\mathrm{HT}}), \tag{4.4.4}$$

where $\widehat{t}_{y,\mathrm{HT}} = \sum_{i\in S} w_i y_i$ denotes the Horvitz–Thompson estimator based on the full sample. The first term represents sampling error, while the second term captures the additional error due to item nonresponse and imputation.

As discussed in Chapter 2, $\mathbb{E}_p(\widehat{t}_{y,\mathrm{HT}} - t_y) = 0$. Since $\widehat{t}_{y,\mathrm{HT}}$ does not depend on the response mechanism or the imputation model, it follows immediately that

$$\mathbb{E}_{mpq}(\widehat{t}_{y,\mathrm{HT}} - t_y) = 0.$$

Consequently, the *mpq* bias of $\widehat{t}_I^R$ is entirely driven by the nonresponse–imputation component:

$$\mathrm{Bias}_{mpq}(\widehat{t}_I^R) = \mathbb{E}_{pq}\Big\{\mathbb{E}_m(\widehat{t}_I^R - \widehat{t}_{y,\mathrm{HT}})\Big\}.$$

Under MAR and the working linear model $\mathbb{E}_m(y_i) = \mathbf{x}_i^\top \boldsymbol{\beta}$, we obtain

$$\mathbb{E}_m(\widehat{t}_I^R - \widehat{t}_{y,\mathrm{HT}}) = -\lambda \left(\sum_{i \in S_m} w_i \mathbf{x}_i \right)^\top (\mathbf{A}_r + \lambda \mathbf{I}_p)^{-1} \boldsymbol{\beta},$$

where

$$\mathbf{A}_r = \sum_{j \in S_r} w_j \mathbf{x}_j \mathbf{x}_j^\top.$$

Therefore,

$$\mathrm{Bias}_{mpq}(\widehat{t}_I^R) = -\lambda \, \mathbb{E}_{pq} \left[\left(\sum_{i \in S_m} w_i \mathbf{x}_i \right)^\top (\mathbf{A}_r + \lambda \mathbf{I}_p)^{-1} \right] \boldsymbol{\beta}. \tag{4.4.5}$$

As expected, the bias vanishes in the case of complete response ($S_m = \emptyset$) or when $\lambda = 0$.

To better understand the behavior of (4.4.5), we consider two simple respondent–design configurations that parallel those used to study ridge estimation of $\boldsymbol{\beta}$ in Section 3.4. Although these configurations are not very realistic in practice, they help illustrate the main ideas.

(1) *Weighted orthonormal respondent design.* Assume

$$\sum_{i \in S_r} w_i \mathbf{x}_i \mathbf{x}_i^\top = \mathbf{I}_p.$$

Then,

$$\mathrm{Bias}_{mpq}(\widehat{t}_I^R) = -\frac{\lambda}{1+\lambda} \sum_{i \in U} w_i (1 - p_i) \mathbf{x}_i^\top \boldsymbol{\beta}.$$

(2) *Weighted standardized uncorrelated respondent design.* Suppose that the predictors are weighted–uncorrelated and standardized within S_r, in the sense that

$$\sum_{i \in S_r} w_i \mathbf{x}_i \mathbf{x}_i^\top = c_r \, \mathbf{I}_p, \qquad c_r = \frac{1}{p} \sum_{i \in S_r} w_i \, \mathbf{x}_i^\top \mathbf{x}_i > 0,$$

so that all weighted cross–moments vanish and each predictor has the same weighted second moment. Then

$$\mathrm{Bias}_{mpq}(\widehat{t}_I^R) = -\mathbb{E}_{pq} \left\{ \frac{\lambda}{c_r + \lambda} \left(\sum_{i \in S_m} w_i \mathbf{x}_i \right)^\top \right\} \boldsymbol{\beta}.$$

For fixed λ, the factor $\lambda/(c_r + \lambda)$ decreases as c_r increases. Thus, all else being equal, the bias is attenuated when c_r is larger, reflecting the fact that c_r grows with the amount of information contributed by the respondent sample.

Ridge imputation generally induces a non-negligible bias due to coefficient shrinkage. To mitigate this effect, we now consider debiased versions of the ridge estimator. The first-order debiased ridge estimator fitted on the respondent sample is defined as

$$\widehat{\boldsymbol{\beta}}_r^{DR}(\lambda) = \left\{\mathbf{I}_p + \lambda\left(\mathbf{A}_r + \lambda\mathbf{I}_p\right)^{-1}\right\}\left(\mathbf{A}_r + \lambda\mathbf{I}_p\right)^{-1}\sum_{i\in S_r} w_i\mathbf{x}_i y_i. \tag{4.4.6}$$

More generally, an mth-order debiased ridge estimator is defined by

$$\widehat{\boldsymbol{\beta}}_{DR}^{(m)}(\lambda) = \left(\sum_{j=0}^{m}\lambda^j\left(\mathbf{A}_r + \lambda\mathbf{I}_p\right)^{-(j+1)}\right)\sum_{i\in S_r} w_i\mathbf{x}_i y_i. \tag{4.4.7}$$

The associated mth-order debiased ridge imputed estimator of t_y is

$$\widehat{t}_I^{DR,(m)} = \sum_{i\in S} w_i r_i y_i + \sum_{i\in S} w_i(1-r_i)\mathbf{x}_i^\top\widehat{\boldsymbol{\beta}}_{DR}^{(m)}(\lambda). \tag{4.4.8}$$

The case $m = 0$ corresponds to standard ridge imputation, while $m = 1$ yields the first-order debiased procedure. Proceeding as before, the mpq bias of $\widehat{t}_I^{DR,(m)}$ is given by

$$\mathrm{Bias}_{mpq}(\widehat{t}_I^{DR,(m)}) = -\lambda^{m+1}\mathbb{E}_{pq}\left[\left(\sum_{i\in S_m} w_i\mathbf{x}_i\right)^\top\left(\mathbf{A}_r + \lambda\mathbf{I}_p\right)^{-(m+1)}\right]\boldsymbol{\beta}. \tag{4.4.9}$$

To better understand the behavior of the mpq bias of the mth-order debiased ridge imputed estimator (including the first-order case $m = 1$), we consider the same two respondent–design configurations as in the ridge imputation case.

(1) *Weighted orthonormal respondent design.* Assume that

$$\sum_{i\in S_r} w_i\,\mathbf{x}_i\mathbf{x}_i^\top = \mathbf{I}_p, \qquad \text{i.e.,} \qquad \mathbf{A}_r = \mathbf{I}_p.$$

Then, (4.4.9) reduces to

$$\mathrm{Bias}_{mpq}\left(\widehat{t}_I^{DR,(m)}\right) = -\frac{\lambda^{m+1}}{(1+\lambda)^{m+1}}\sum_{i\in U} w_i(1-p_i)\mathbf{x}_i^\top\boldsymbol{\beta} \tag{4.4.10}$$

(2) *Weighted standardized uncorrelated respondent design.* Assume that the predictors are weighted–uncorrelated and standardized within S_r so that

$$\sum_{i\in S_r} w_i\,\mathbf{x}_i\mathbf{x}_i^\top = c_r\,\mathbf{I}_p, \qquad c_r > 0, \qquad \text{i.e.,} \qquad \mathbf{A}_r = c_r\,\mathbf{I}_p.$$

Then, (4.4.9) simplifies to

$$\mathrm{Bias}_{mpq}\left(\widehat{t}_I^{DR,(m)}\right) = -\mathbb{E}_{pq}\left[\frac{\lambda^{m+1}}{(c_r+\lambda)^{m+1}}\left(\sum_{i\in S_m} w_i\mathbf{x}_i\right)^\top\boldsymbol{\beta}\right]. \tag{4.4.11}$$

For fixed λ, the magnitude of the mpq bias decreases as c_r increases. For fixed $\lambda > 0$ and $c_r > 0$, the multiplicative factor

$$\left(\frac{\lambda}{c_r + \lambda}\right)^{m+1}$$

satisfies $0 < \lambda/(c_r + \lambda) < 1$ and therefore decreases geometrically as m increases.

4.5 Conditional MSE for Ridge Imputation

In ridge and LASSO regression, the choice of the tuning parameter λ is an important step. In practice, a common approach is to select λ using K-fold cross-validation, which aims at finding the value that gives the best prediction performance. However, in survey sampling, the problem is not one of prediction but of estimation. The goal is not to predict individual outcomes, but rather to estimate a finite population parameter, such as a total or a mean. From this perspective, selecting λ based on prediction accuracy may not be appropriate. Instead, we would like to choose the value of λ that minimizes the mean squared error of the estimator of interest. Since the true mean squared error is unknown, it must be estimated. Therefore, in this thesis, we select λ by minimizing an estimate of the mean squared error. More specifically, we use the conditional mean squared error (CMSE) as our measure of performance. The CMSE and its estimation are described in the next sections.

4.5.1 Decomposition of the Nonresponse Error

To simplify the algebra, we work in the remainder of this section with the population mean $\mu = t_y/N$ rather than the population total t_y. All results carry over immediately to totals, up to the usual scaling factors $1/N$ for point estimation and $1/N^2$ for variance.

The CMSE is designed to isolate the error introduced by nonresponse and imputation. Since the sampling error is already captured by the unbiased Horvitz–Thompson estimator and is not affected by the imputation procedure, the CMSE is defined by comparing the imputed estimator

$$\widehat{\mu}_I = \frac{1}{N}\sum_{i\in S} w_i \widetilde{y}_i, \qquad \widetilde{y}_i = r_i y_i + (1 - r_i) y_i^*,$$

to the Horvitz–Thompson estimator of μ,

$$\widehat{\mu}_{HT} = \frac{1}{N}\sum_{i\in S} w_i y_i,$$

with expectation $\mathbb{E}_m(\cdot)$ taken with respect to the imputation model. The CMSE is then defined as

$$\mathrm{CMSE}(\widehat{\mu}_I) = \mathbb{E}_m\big[(\widehat{\mu}_I - \widehat{\mu}_{HT})^2\big]$$

$$= \mathbb{V}_m(\widehat{\mu}_I - \widehat{\mu}_{HT}) + \{\mathbb{E}_m(\widehat{\mu}_I - \widehat{\mu}_{HT})\}^2 . \tag{4.5.1}$$

The first term corresponds to the variance induced by nonresponse and imputation, while the second term represents the squared nonresponse bias.

We start from the identity

$$\widehat{\mu}_I - \widehat{\mu}_{HT} = -\frac{1}{N}\sum_{i\in S_m} w_i(y_i - y_i^*), \tag{4.5.2}$$

where y_i^* denotes the imputed value used to replace the missing y_i.

We now consider linear regression imputation with an intercept. The imputation model is written as

$$y_i = \mathbf{x}_i^\top \boldsymbol{\beta} + \epsilon_i, \qquad \mathbf{x}_i = \begin{pmatrix} 1 \\ \mathbf{z}_i \end{pmatrix}, \qquad \boldsymbol{\beta} = \begin{pmatrix} \alpha \\ \boldsymbol{\gamma} \end{pmatrix},$$

where the first component of $\mathbf{x}_i$ corresponds to the intercept. The parameter vector $\boldsymbol{\beta}$ is estimated using the respondent sample S_r, yielding an estimator $\widehat{\boldsymbol{\beta}}$ that may correspond to ordinary least squares, ridge regression, or a debiased ridge estimator.

The decomposition derived below holds regardless of the specific estimation method used for $\boldsymbol{\beta}$. This is because these estimators can all be written as linear functions of the observed responses, which implies that the resulting imputation estimators are linear in the data.

Under regression imputation, $y_i^* = \mathbf{x}_i^\top \widehat{\boldsymbol{\beta}}$, and substituting into (4.5.2) yields the exact identity

$$\widehat{\mu}_I - \widehat{\mu}_{HT} = \frac{1}{N}\sum_{i\in S_m} w_i \mathbf{x}_i^\top (\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}) - \frac{1}{N}\sum_{i\in S_m} w_i \epsilon_i. \tag{4.5.3}$$

Define the weighted totals

$$\widehat{N}_r = \sum_{i\in S_r} w_i, \qquad \widehat{N}_m = \sum_{i\in S_m} w_i,$$

and the corresponding weighted mean covariate vectors

$$\overline{\mathbf{x}}_r = \widehat{N}_r^{-1}\sum_{i\in S_r} w_i \mathbf{x}_i, \qquad \overline{\mathbf{x}}_m = \widehat{N}_m^{-1}\sum_{i\in S_m} w_i \mathbf{x}_i, \qquad \Delta_{\mathbf{x}} = \bar{\mathbf{x}}_m - \bar{\mathbf{x}}_r.$$

Because the first component of $\mathbf{x}_i$ equals one for all units, the intercept cancels in $\Delta_{\mathbf{x}}$.

Using the normal equations associated with estimation of the intercept on the respondent sample, which hold regardless of whether $\widehat{\boldsymbol{\beta}}$ is obtained by least squares, ridge regression, or a debiased ridge procedure (provided the intercept is not penalized), expression (4.5.3) can be rewritten exactly as

$$\widehat{\mu}_I - \widehat{\mu}_{HT} = \frac{\widehat{N}_m}{N}\left\{\Delta_{\mathbf{x}}^\top(\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}) + (\overline{\epsilon}_r - \overline{\epsilon}_m)\right\}, \tag{4.5.4}$$

where

$$\bar{\epsilon}_r = \widehat{N}_r^{-1} \sum_{i \in S_r} w_i \epsilon_i, \qquad \bar{\epsilon}_m = \widehat{N}_m^{-1} \sum_{i \in S_m} w_i \epsilon_i.$$

The decomposition in (4.5.4) is purely algebraic and does not depend on the specific choice of $\widehat{\boldsymbol{\beta}}$. Different estimators affect the magnitude of the bias and variance components of the CMSE, but not the structure of the decomposition itself. In the subsequent subsection, we exploit this representation to derive the CMSE associated with ridge imputation, first-order debiased ridge imputation, and the mth-order debiasing scheme.

4.5.2 Derivation of the CMSE

Before analyzing specific estimators, we establish the general forms of the conditional bias and variance based on the decomposition in (4.5.4). These general results apply to any linear estimator $\widehat{\boldsymbol{\beta}}$ used for imputation.

First, we consider the conditional bias. Taking the model expectation $\mathbb{E}_m$ of (4.5.4), and noting that $\mathbb{E}_m(\epsilon_i) = 0$ implies $\mathbb{E}_m(\bar{\epsilon}_r - \bar{\epsilon}_m) = 0$, the conditional bias of $\widehat{\mu}_I$ is given by

$$\text{Bias}(\widehat{\mu}_I) = \mathbb{E}_m(\widehat{\mu}_I - \widehat{\mu}_{HT}) = \frac{\widehat{N}_m}{N} \Delta_{\mathbf{x}}^\top \mathbb{E}_m(\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}). \tag{4.5.5}$$

Next, the conditional variance is given by

$$\begin{aligned}
\mathbb{V}_m(\widehat{\mu}_I - \widehat{\mu}_{HT}) &= \left(\frac{\widehat{N}_m}{N}\right)^2 \mathbb{V}_m\Big\{\Delta_{\mathbf{x}}^\top(\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}) + (\bar{\epsilon}_r - \bar{\epsilon}_m)\Big\} \\
&= \left(\frac{\widehat{N}_m}{N}\right)^2 \Big[\Delta_{\mathbf{x}}^\top \mathbb{V}_m(\widehat{\boldsymbol{\beta}}) \Delta_{\mathbf{x}} + \mathbb{V}_m(\bar{\epsilon}_r - \bar{\epsilon}_m) + 2\,\text{Cov}_m\big(\Delta_{\mathbf{x}}^\top \widehat{\boldsymbol{\beta}},\, \bar{\epsilon}_r\big)\Big].
\end{aligned} \tag{4.5.6}$$

Note that

$$\begin{aligned}
\mathbb{V}_m(\bar{\epsilon}_r - \bar{\epsilon}_m) &= \mathbb{V}_m(\bar{\epsilon}_r) + \mathbb{V}_m(\bar{\epsilon}_m) \\
&= \mathbb{V}_m\left(\frac{\sum_{i \in S} w_i r_i \epsilon_i}{\widehat{N}_r}\right) + \mathbb{V}_m\left(\frac{\sum_{i \in S} w_i (1 - r_i) \epsilon_i}{\widehat{N}_m}\right) \\
&= \sigma^2 \left(\frac{\sum_{i \in S} w_i^2 r_i}{\widehat{N}_r^2} + \frac{\sum_{i \in S} w_i^2 (1 - r_i)}{\widehat{N}_m^2}\right).
\end{aligned} \tag{4.5.7}$$

Under simple random sampling without replacement, whereby $w_i = N/n$, Expression (4.5.7) reduces to

$$\mathbb{V}_m(\bar{\epsilon}_r - \bar{\epsilon}_m) = \sigma^2 \left(\frac{1}{n_r} + \frac{1}{n_m}\right).$$

4.5.3 CMSE for Ridge Imputation

In this section, we derive an expression of the CMSE in the case of ridge imputation (which corresponds to $m = 0$). Let

$$\mathbf{A}(\lambda) = \sum_{i \in S_r} w_i \mathbf{x}_i \mathbf{x}_i^\top + \lambda \mathbf{I}_p.$$

Therefore, the squared conditional bias is given by

$$\mathbb{E}_m\Big\{\widehat{\boldsymbol{\beta}}_R(\lambda) - \boldsymbol{\beta}\Big\} = -\lambda \mathbf{A}(\lambda)^{-1}\boldsymbol{\beta}.$$

Substituting into (4.5.5), the squared bias component is

$$\text{Bias}^2(\widehat{\mu}_I^{(0)}) = \left(\frac{\widehat{N}_m}{N}\right)^2 \lambda^2 \left(\Delta_{\mathbf{x}}^\top \mathbf{A}(\lambda)^{-1}\boldsymbol{\beta}\right)^2. \tag{4.5.8}$$

The variance of the ridge coefficients is

$$\mathbb{V}_m\Big(\widehat{\boldsymbol{\beta}}_R(\lambda)\Big) = \sigma^2\, \mathbf{A}(\lambda)^{-1} \left(\sum_{i \in S_r} w_i^2 \mathbf{x}_i \mathbf{x}_i^\top\right) \mathbf{A}(\lambda)^{-1}.$$

The covariance term appearing in (4.5.6) is

$$\text{Cov}_m\big(\Delta_{\mathbf{x}}^\top \widehat{\boldsymbol{\beta}}_R(\lambda), \bar{\epsilon}_r\big) = \frac{\sigma^2}{\widehat{N}_r}\, \Delta_{\mathbf{x}}^\top \mathbf{A}(\lambda)^{-1} \left(\sum_{i \in S_r} w_i^2 \mathbf{x}_i\right).$$

Combining these results, the CMSE for ridge imputation is

$$\begin{aligned}\text{CMSE}^{(0)}(\lambda) = \left(\frac{\widehat{N}_m}{N}\right)^2 \Bigg[& \lambda^2 \left(\Delta_{\mathbf{x}}^\top \mathbf{A}(\lambda)^{-1}\boldsymbol{\beta}\right)^2 + \sigma^2 \Delta_{\mathbf{x}}^\top \mathbf{A}(\lambda)^{-1} \left(\sum_{i \in S_r} w_i^2 \mathbf{x}_i \mathbf{x}_i^\top\right) \mathbf{A}(\lambda)^{-1} \Delta_{\mathbf{x}} \\ & + \sigma^2 \left(\frac{\sum_{i \in S} w_i^2 r_i}{\widehat{N}_r^2} + \frac{\sum_{i \in S} w_i^2 (1 - r_i)}{\widehat{N}_m^2}\right) + \frac{2\sigma^2}{\widehat{N}_r}\, \Delta_{\mathbf{x}}^\top \mathbf{A}(\lambda)^{-1} \left(\sum_{i \in S_r} w_i^2 \mathbf{x}_i\right)\Bigg].\end{aligned} \tag{4.5.9}$$

4.5.4 CMSE for First-Order Debiased Ridge Imputation

In this section, we derive an expression of the CMSE in the case of the first-order debiased ridge imputation (which corresponds to $m = 1$). We have

$$\mathbb{E}_m\Big\{\widehat{\boldsymbol{\beta}}_{DR}^{(1)}(\lambda) - \boldsymbol{\beta}\Big\} = \lambda^2 \mathbf{A}(\lambda)^{-2}\boldsymbol{\beta}.$$

Using the general bias formula (4.5.5), the squared bias component becomes

$$\text{Bias}^2(\widehat{\mu}_I^{(1)}) = \left(\frac{\widehat{N}_m}{N}\right)^2 \lambda^4 \left(\Delta_{\mathbf{x}}^\top \mathbf{A}(\lambda)^{-2}\boldsymbol{\beta}\right)^2. \tag{4.5.10}$$

The variance of the first-order debiased ridge coefficients is

$$\mathbb{V}_m\Big(\widehat{\boldsymbol{\beta}}_{DR}^{(1)}(\lambda)\Big) = \sigma^2\,\mathbf{D}_1(\lambda)\left(\sum_{i\in S_r} w_i^2\mathbf{x}_i\mathbf{x}_i^\top\right)\mathbf{D}_1(\lambda)^\top, \qquad \mathbf{D}_1(\lambda) = \mathbf{A}(\lambda)^{-1} + \lambda\mathbf{A}(\lambda)^{-2}.$$

The covariance term in (4.5.6) is

$$\mathrm{Cov}_m\big(\Delta_{\mathbf{x}}^\top\widehat{\boldsymbol{\beta}}_{DR}^{(1)}(\lambda), \overline{\epsilon}_r\big) = \frac{\sigma^2}{\widehat{N}_r}\,\Delta_{\mathbf{x}}^\top\mathbf{D}_1(\lambda)\left(\sum_{i\in S_r} w_i^2\mathbf{x}_i\right).$$

Combining these results, the CMSE for first-order debiased ridge imputation is

$$\begin{aligned}\mathrm{CMSE}^{(1)}(\lambda) = \left(\frac{\widehat{N}_m}{N}\right)^2 \Bigg[&\lambda^4\left(\Delta_{\mathbf{x}}^\top\mathbf{A}(\lambda)^{-2}\boldsymbol{\beta}\right)^2 + \sigma^2\Delta_{\mathbf{x}}^\top\mathbf{D}_1(\lambda)\left(\sum_{i\in S_r} w_i^2\mathbf{x}_i\mathbf{x}_i^\top\right)\mathbf{D}_1(\lambda)^\top\Delta_{\mathbf{x}} \\ &+ \sigma^2\left(\frac{\sum_{i\in S} w_i^2 r_i}{\widehat{N}_r^2} + \frac{\sum_{i\in S} w_i^2(1-r_i)}{\widehat{N}_m^2}\right) + \frac{2\sigma^2}{\widehat{N}_r}\,\Delta_{\mathbf{x}}^\top\mathbf{D}_1(\lambda)\left(\sum_{i\in S_r} w_i^2\mathbf{x}_i\right)\Bigg].\end{aligned} \tag{4.5.11}$$

4.5.5 CMSE for m-th Order Debiased Ridge Imputation

In this section, we generalize the previous results to the mth-order debiased ridge estimator. First, we have $\widehat{\boldsymbol{\beta}}_{DR}^{(m)}(\lambda)$ is

$$\mathbb{E}_m\Big\{\widehat{\boldsymbol{\beta}}_{DR}^{(m)}(\lambda) - \boldsymbol{\beta}\Big\} = (-1)^{m+1}\lambda^{m+1}\mathbf{A}(\lambda)^{-(m+1)}\boldsymbol{\beta}, \tag{4.5.12}$$

It follows that the squared bias component of the imputed estimator is

$$\mathrm{Bias}^2(\widehat{\mu}_I^{(m)}) = \left(\frac{\widehat{N}_m}{N}\right)^2\lambda^{2(m+1)}\left(\Delta_{\mathbf{x}}^\top\mathbf{A}(\lambda)^{-(m+1)}\boldsymbol{\beta}\right)^2. \tag{4.5.13}$$

Next, we derive the conditional variance. From (4.4.7), we define

$$\mathbf{D}_m(\lambda) = \sum_{j=0}^{m}\lambda^j\mathbf{A}(\lambda)^{-(j+1)},$$

so that

$$\widehat{\boldsymbol{\beta}}_{DR}^{(m)}(\lambda) = \mathbf{D}_m(\lambda)\sum_{i\in S_r} w_i\mathbf{x}_i y_i.$$

The variance of the coefficient estimator is

$$\mathbb{V}_m\Big(\widehat{\boldsymbol{\beta}}_{DR}^{(m)}(\lambda)\Big) = \mathbf{D}_m(\lambda)\mathbb{V}_m\left(\sum_{i\in S_r} w_i\mathbf{x}_i y_i\right)\mathbf{D}_m(\lambda)^\top$$

$$= \sigma^2 \, \mathbf{D}_m(\lambda) \left(\sum_{i \in S_r} w_i^2 \mathbf{x}_i \mathbf{x}_i^\top \right) \mathbf{D}_m(\lambda)^\top. \tag{4.5.14}$$

The covariance term appearing in the general variance decomposition (4.5.6) is

$$\mathrm{Cov}_m\big(\Delta_{\mathbf{x}}^\top \widehat{\boldsymbol{\beta}}_{DR}^{(m)}(\lambda), \bar{\epsilon}_r\big) = \frac{\sigma^2}{\widehat{N}_r} \, \Delta_{\mathbf{x}}^\top \mathbf{D}_m(\lambda) \left(\sum_{i \in S_r} w_i^2 \mathbf{x}_i \right). \tag{4.5.15}$$

Combining these results, the CMSE of the mth-order debiased ridge imputed estimator is

$$\begin{aligned} \mathrm{CMSE}^{(m)}(\lambda) = \left(\frac{\widehat{N}_m}{N} \right)^2 \Bigg[& \lambda^{2(m+1)} \left(\Delta_{\mathbf{x}}^\top \mathbf{A}(\lambda)^{-(m+1)} \boldsymbol{\beta} \right)^2 \\ & + \sigma^2 \Delta_{\mathbf{x}}^\top \mathbf{D}_m(\lambda) \left(\sum_{i \in S_r} w_i^2 \mathbf{x}_i \mathbf{x}_i^\top \right) \mathbf{D}_m(\lambda)^\top \Delta_{\mathbf{x}} \\ & + \sigma^2 \left(\frac{\sum_{i \in S} w_i^2 r_i}{\widehat{N}_r^2} + \frac{\sum_{i \in S} w_i^2 (1 - r_i)}{\widehat{N}_m^2} \right) + \frac{2\sigma^2}{\widehat{N}_r} \, \Delta_{\mathbf{x}}^\top \mathbf{D}_m(\lambda) \left(\sum_{i \in S_r} w_i^2 \mathbf{x}_i \right) \Bigg]. \end{aligned} \tag{4.5.16}$$

The CMSE given in (4.5.16) is not directly available in practice, as it depends on the unknown quantities $\boldsymbol{\beta}$ and σ^2. Therefore, it must be estimated by replacing these quantities with suitable plug-in estimators.

A first option is to estimate $\boldsymbol{\beta}$ and σ^2 using ordinary least squares. In this case, we use

$$\widehat{\boldsymbol{\beta}}_r = \left(\sum_{i \in S_r} w_i \mathbf{x}_i c_i^{-1} \mathbf{x}_i^\top \right)^{-1} \sum_{i \in S_r} w_i \mathbf{x}_i c_i^{-1} y_i,$$

and

$$\widehat{\sigma}^2 = \frac{1}{n_r - p} \sum_{i \in S_r} \left(y_i - \mathbf{x}_i^\top \widehat{\boldsymbol{\beta}}_r \right)^2.$$

An alternative is to use ridge-based estimators. In this case, $\boldsymbol{\beta}$ is replaced by the ridge estimator $\widehat{\boldsymbol{\beta}}_R(\lambda)$, and σ^2 is estimated using the corresponding residuals:

$$\widehat{\sigma}_R^2(\lambda) = \frac{1}{n_r - \mathrm{df}(\lambda)} \sum_{i \in S_r} \left(y_i - \mathbf{x}_i^\top \widehat{\boldsymbol{\beta}}_R(\lambda) \right)^2,$$

where $\mathrm{df}(\lambda) = \mathrm{tr}\{\mathbf{H}(\lambda)\}$ denotes the effective degrees of freedom of the ridge fit with

$$\mathbf{H}(\lambda) = \mathbf{X}_r \big(\mathbf{X}_r^\top \mathbf{X}_r + \lambda \mathbf{I}_p \big)^{-1} \mathbf{X}_r^\top.$$

These two approaches will be compared in the simulation study in Section 4.7.

The estimated CMSE can then be used as a criterion to select the tuning parameters. We consider two strategies.

(i) In the first approach, the order m is fixed in advance, and the regularization parameter λ is selected by minimizing the estimated CMSE over a predefined grid of candidate values. This yields the optimal λ for a given value of m.

(ii) In the second approach, both m and λ are selected jointly by minimizing the estimated CMSE over a two-dimensional grid of candidate pairs (m, λ). This allows the data to determine both the amount of regularization and the level of bias correction.

In both cases, the estimated CMSE directly targets the accuracy of the imputed estimator of the population parameter.

4.6 LASSO and Post-LASSO Imputation

LASSO imputation extends the linear imputation framework by incorporating an L_1 penalty in the estimation of regression coefficients, thereby encouraging sparsity in the fitted model. The motivation parallels that of LASSO regression: when the number of auxiliary variables is large, or when multicollinearity is present, traditional linear or ridge imputation may produce unstable or inefficient estimates. By simultaneously performing variable selection and coefficient shrinkage, LASSO imputation offers a more interpretable model and can help mitigate overfitting, particularly in high-dimensional settings. In this section, we formalize the LASSO imputation approach and discuss its estimation procedure, as well as a common two-step extension known as Post-LASSO.

Let $\widehat{\boldsymbol{\beta}}_r^{LA}(\lambda)$ represent the LASSO estimator obtained from the respondent sample. Then, the imputed value for unit $i \in S_m$ is given by

$$y_{i,LA}^* = \mathbf{x}_i^\top \widehat{\boldsymbol{\beta}}_r^{LA}(\lambda), i \in S_m, \tag{4.6.1}$$

The corresponding imputed total estimator based on LASSO follows the standard imputation formula:

$$\widehat{t}_I^{LA} = \sum_{i \in S} w_i r_i y_i + \sum_{i \in S} w_i (1 - r_i) y_{i,LA}^*. \tag{4.6.2}$$

While LASSO is effective for automatic variable selection, it tends to introduce bias in the estimated coefficients, particularly for those with large true values. To reduce this bias, a two-step procedure known as Post-LASSO is often employed. In this approach, the variables with nonzero coefficients identified by the initial LASSO fit are retained, and an OLS regression is subsequently applied to this selected subset. The resulting estimator, denoted by $\widehat{\boldsymbol{\beta}}_r^{PL}$, is then used to compute the imputed values as follows:

$$y_{i,PL}^* = \mathbf{x}_i^\top \widehat{\boldsymbol{\beta}}_r^{PL}, i \in S_m. \tag{4.6.3}$$

The corresponding imputed total estimator is given by:

$$\widehat{t}_I^{PL} = \sum_{i \in S} w_i r_i y_i + \sum_{i \in S} w_i (1 - r_i) y_{i,PL}^*. \tag{4.6.4}$$

In practice, Post-LASSO can reduce the bias introduced by the LASSO, since the final OLS step is not penalized. At the same time, it keeps the variable selection obtained in the first step. As a result, it can provide more accurate estimates, especially when some relevant variables have been correctly selected by LASSO.

In the next section, we present a simulation study to evaluate the performance of several imputation estimators, including ridge-based methods (and their debiased versions), as well as Adaptive LASSO and Post-LASSO. The simulations consider different scenarios, such as varying levels of correlation, signal strength, and missingness, in order to assess how these methods perform in practice.

4.7 Empirical results

4.7.1 Simulation Setup

In this simulation experiment, the data generation process for the design matrix $\mathbf{X}$ and the response variable $\mathbf{y}$ follows the setup described in Section 2.3.4. We consider different values of p, namely $p \in \{10, 20, 30, 40, 50\}$. For each value of p, each row $\mathbf{x}_i$ of the design matrix $\mathbf{X}$ is generated from a multivariate normal distribution with mean vector $\mathbf{0}$ and covariance matrix having ones on the diagonal and ρ on all off-diagonal entries. We consider two levels of correlation between the covariates, namely $\rho = 0.3$ and $\rho = 0.95$.

We consider three types of models corresponding to different levels of sparsity: a non-sparse model (no zero coefficients), a moderately sparse model with 40% of the coefficients equal to zero, and a highly sparse model with 60% of the coefficients equal to zero.

In the non-sparse case, all regression coefficients are nonzero and are generated as

$$\beta_j \sim \begin{cases} \text{Unif}(0,1), & j = 1, \ldots, \lfloor p/2 \rfloor, \\ \text{Unif}(1,2), & j = \lfloor p/2 \rfloor + 1, \ldots, p. \end{cases} \tag{4.7.1}$$

For the sparse models, a proportion of the coefficients is set to zero (40% or 60%, depending on the scenario). The remaining nonzero coefficients are generated in the same way as in (4.7.1), by splitting them into two groups and sampling from $\text{Unif}(0,1)$ and $\text{Unif}(1,2)$.

For each simulation setting, the covariates $\mathbf{X}$ and the regression coefficients $\boldsymbol{\beta}$ were generated once and then kept fixed throughout the experiment. The number of predictors was taken to be

$$p \in \{10, 20, 30, 40, 50\}.$$

Given these fixed values of $\mathbf{X}$ and $\boldsymbol{\beta}$, we then generated $R = 5000$ independent realizations of the response variable Y. This can be viewed as generating 5000 finite populations, each of size $N = 2000$ under the same underlying regression structure.

More precisely, the response variable was generated according to

$$y_i \;=\; 10 + \mathbf{x}_i^{\mathsf{T}}\boldsymbol{\beta} + \varepsilon_i, \qquad \varepsilon_i \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma^2),$$

where σ^2 was chosen so that the resulting coefficient of determination R^2 was equal to 0.6.

For each of the 5000 finite populations, a simple random sample without replacement of size $n = 200$ was selected.

Next, nonresponse was generated within each sample according to a Missing at Random (MAR) mechanism. Specifically, the response probability for unit i was given by

$$p_i = 0.1 + 0.9\,\frac{1}{1 + \exp\bigl[-(\gamma_0 + \mathbf{x}_i^{\top}\boldsymbol{\gamma})\bigr]},$$

where $\boldsymbol{\gamma}$ is a vector whose components are all equal to 0.2, and the intercept γ_0 was chosen so that the overall response rate was approximately 50%.

For each incomplete sample, missing values were imputed and the corresponding imputed estimators were computed:

(i) the full-sample Horvitz–Thompson estimator, computed using the complete sample before nonresponse was introduced, as a benchmark;

(ii) the imputed estimator based on ordinary least squares (OLS);

(iii) the imputed estimator based on ridge regression;

(iv) the imputed estimator based on the mth-order debiased ridge estimator, for $m = 0, \ldots, 5$;

(v) the imputed estimator based on the debiased ridge estimator with (m, λ) selected by minimizing the estimated CMSE;

(vi) the imputed estimator based on the Adaptive LASSO;

(vii) the imputed estimator based on the Post-LASSO.

The simulation study is divided into two parts, each addressing a different objective. In the first part, we focus on the estimation of the conditional mean squared error (CMSE) for ridge-based imputation methods, comparing their MSE to investigate how different plug-in choices affect its accuracy. In the second part, we compare the performance of several imputation methods, including ridge-based estimators, Adaptive LASSO, and Post-LASSO, and we also examine the impact of different tuning parameter selection strategies.

4.7.2 Evaluation of CMSE estimation

In a first set of experiments, we focus on the estimation of the conditional mean squared error (CMSE) for ridge-based imputation methods, including both standard and debiased ridge imputation. The goal is to assess how different plug-in choices for $\boldsymbol{\beta}$ and σ^2 affect the quality of the CMSE approximation.

In particular, we compare three approaches:

(I) CMSE based on OLS estimators of $\boldsymbol{\beta}$ and σ^2; see Section 4.5.5.

(II) CMSE based on ridge estimators of $\boldsymbol{\beta}$ and σ^2; see Section 4.5.5.

(III) an oracle version, where the true values of $\boldsymbol{\beta}$ and σ^2 are used. The oracle case is infeasible in practice but serves as a benchmark,

The results are reported in Tables 4.3-4.6.

We use DR-m (CMSE-R/CMSE-OLS/CMSE-Oracle) to denote the mth order debiaed ridge imputation based on ridge/OLS/Oracle estimation of $\boldsymbol{\beta}$ and σ^2, respectively. We use BEST(CMSE-R/CMSE-OLS/CMSE-Oracle) to denote the fully data-driven approach in which both m and λ are selected jointly by minimizing the estimated CMSE over a two-dimensional grid of candidate values.

Across all four tables, a clear pattern emerges regarding the estimation of the CMSE. As expected, the oracle version consistently provides the best results, serving as a gold standard. It typically achieves both the smallest bias and the lowest MSE. For instance, in the high-correlation setting without sparsity (Table 4.4), the oracle-based estimators attain MSE values around 14.4–14.6 when $P/E(n_r) = 30/100$, which are uniformly smaller than their OLS-based counterparts (around 16.0) and comparable or slightly better than the ridge-based ones.

Among feasible approaches, using ridge-based plug-in estimators for $\boldsymbol{\beta}$ and σ^2 generally leads to better performance in terms of MSE than using OLS. This is particularly evident in more challenging settings. For example, in Table 4.4 with $P/E(n_r) = 50/100$, the MSE for CMSE-OLS is around 51.8, whereas it is about 41.1 for CMSE-R, showing a substantial improvement when ridge is used. On the other hand, OLS-based estimators often exhibit slightly smaller bias than ridge-based ones, especially for higher-order debiased estimators. This can be seen, for instance, in Table 4.3, where the relative bias for DR-5 is about 0.1 under CMSE-OLS compared to 0.1–0.3 under CMSE-R, although the corresponding MSE remains very similar or slightly larger under OLS. This illustrates a classical bias–variance trade-off: OLS tends to reduce bias but at the cost of increased variability, whereas ridge introduces a small amount of bias but stabilizes the estimation, leading to lower MSE overall. This effect becomes more pronounced as the problem becomes more ill-conditioned, such as

when the correlation is high or when $P/E(n_r)$ increases.

Finally, these conclusions are consistent across all scenarios, including both low and high correlation as well as sparse and non-sparse settings (Tables 4.3–4.6), confirming that ridge-based plug-in estimators provide a good practical compromise, while the oracle benchmark remains unattainable in practice.

4.7.3 Comparison of several imputation methods

In a second set of experiments, we compare ridge-based imputation methods with alternative approaches, including Adaptive LASSO and Post-LASSO. In this part, we also investigate the impact of the tuning parameter selection method. In particular, ridge and debiased ridge estimators are considered with tuning parameters selected either by cross-validation (CV) or by minimizing the estimated CMSE using $\hat{\boldsymbol{\beta}}$ and σ^2 estimated through ridge.

More precisely, for the debiased ridge estimators, we consider orders $m = 0, \ldots, 5$, where the case $m = 0$ corresponds to the standard ridge estimator. For each value of m, the tuning parameter λ is selected either by CV or by minimizing the estimated CMSE. This leads to the estimators denoted by DR-m (CV) and DR-m (CMSE), respectively.

In addition, we consider a fully data-driven approach in which both m and λ are selected jointly by minimizing the estimated CMSE over a two-dimensional grid of candidate values. This yields the estimator denoted by BEST (CMSE).

We examine the following imputation estimators:

(i) the full-sample Horvitz–Thompson estimator (denoted Full), computed using the complete sample before nonresponse, as a benchmark;

(ii) the imputed estimator based on ordinary least squares (OLS);

(iii) the imputed estimator based on ridge and debiased ridge regression, for $m = 0, \ldots, 5$, with λ selected either by cross-validation (DR-m (CV)) or by minimizing the estimated CMSE (DR-m (CMSE));

(iv) the imputed estimator based on the debiased ridge estimator with both m and λ selected jointly by minimizing the estimated CMSE (BEST (CMSE));

(v) the imputed estimator based on the Adaptive LASSO (Adaptive LASSO (CV));

(vi) the imputed estimator based on the Post-LASSO (Post-LASSO).

Table 4.3: Comparison of different estimators of CMSE under no sparsity at corr = 0.3, showing % Relative Bias and MSE in brackets.

Estimator	$P/E(n_r)$				
	$\frac{10}{100}$	$\frac{20}{100}$	$\frac{30}{100}$	$\frac{40}{100}$	$\frac{50}{100}$
Ridge (CMSE)	0.7 (0.54)	2.5 (2.84)	4.4 (5.57)	7.0 (10.04)	9.8 (16.34)
DR-1 (CMSE-R)	0.5 (0.57)	2.0 (2.72)	3.3 (5.09)	4.3 (8.94)	5.1 (14.62)
DR-2 (CMSE-R)	0.2 (0.57)	0.9 (2.68)	1.4 (5.12)	2.1 (9.18)	2.8 (15.21)
DR-3 (CMSE-R)	0.1 (0.57)	0.4 (2.72)	0.8 (5.24)	1.4 (9.49)	2.0 (15.83)
DR-4 (CMSE-R)	0.1 (0.57)	0.2 (2.75)	0.5 (5.35)	1.0 (9.75)	1.6 (16.36)
DR-5 (CMSE-R)	0.1 (0.57)	0.2 (2.78)	0.3 (5.45)	0.8 (9.98)	1.3 (16.83)
BEST (CMSE-R)	0.5 (0.58)	2.2 (2.77)	3.9 (5.29)	5.8 (9.52)	7.8 (15.68)
DR-1 (CMSE-OLS)	0.6 (0.58)	1.5 (2.75)	1.7 (5.33)	2.1 (9.90)	2.2 (17.34)
DR-2 (CMSE-OLS)	0.4 (0.57)	0.7 (2.71)	0.8 (5.30)	1.2 (9.94)	1.4 (17.49)
DR-3 (CMSE-OLS)	0.2 (0.57)	0.4 (2.72)	0.5 (5.34)	0.9 (10.03)	1.1 (17.69)
DR-4 (CMSE-OLS)	0.1 (0.57)	0.3 (2.73)	0.4 (5.40)	0.7 (10.14)	1.0 (17.87)
DR-5 (CMSE-OLS)	0.1 (0.57)	0.2 (2.75)	0.3 (5.45)	0.6 (10.23)	0.8 (18.06)
BEST (CMSE-OLS)	0.5 (0.57)	1.6 (2.83)	2.7 (5.67)	4.3 (10.59)	5.4 (18.30)
DR-1 (CMSE-Oracle)	0.6 (0.56)	2.1 (2.64)	3.4 (5.02)	4.4 (8.90)	5.2 (14.58)
DR-2 (CMSE-Oracle)	0.5 (0.56)	1.0 (2.65)	1.5 (5.10)	2.1 (9.17)	2.8 (15.21)
DR-3 (CMSE-Oracle)	0.3 (0.56)	0.5 (2.69)	0.8 (5.23)	1.4 (9.48)	2.0 (15.82)
DR-4 (CMSE-Oracle)	0.2 (0.56)	0.3 (2.72)	0.5 (5.33)	1.1 (9.73)	1.6 (16.34)
DR-5 (CMSE-Oracle)	0.1 (0.57)	0.2 (2.74)	0.3 (5.41)	0.8 (9.94)	1.3 (16.79)
BEST (CMSE-Oracle)	0.5 (0.56)	1.8 (2.64)	3.3 (5.02)	4.4 (8.90)	5.2 (14.59)

Table 4.4: Comparison of different estimators of CMSE under no sparsity at corr = 0.95, showing % Relative Bias and MSE in brackets.

Estimator	$P/E(n_r)$				
	$\frac{10}{100}$	$\frac{20}{100}$	$\frac{30}{100}$	$\frac{40}{100}$	$\frac{50}{100}$
Ridge (CMSE)	0.5 (1.56)	1.1 (7.54)	1.2 (14.58)	2.1 (26.22)	39.9 (41.05)
DR-1 (CMSE-R)	-0.1 (1.56)	-0.2 (7.53)	-0.5 (14.56)	0.2 (26.28)	2.1 (41.15)
DR-2 (CMSE-R)	-0.2 (1.56)	-0.5 (7.53)	-0.6 (14.54)	0.1 (26.23)	2.0 (41.07)
DR-3 (CMSE-R)	-0.2 (1.56)	-0.5 (7.53)	-0.6 (14.53)	0.1 (26.22)	1.9 (41.01)
DR-4 (CMSE-R)	-0.2 (1.56)	-0.5 (7.53)	-0.6 (14.54)	0.1 (26.26)	1.9 (41.08)
DR-5 (CMSE-R)	-0.2 (1.56)	-0.5 (7.54)	-0.6 (14.56)	0.0 (26.34)	1.9 (41.21)
BEST (CMSE-R)	0.4 (1.56)	0.9 (7.55)	0.9 (14.59)	1.8 (26.26)	3.6 (41.11)
DR-1 (CMSE-OLS)	-0.0 (1.58)	-0.2 (7.94)	-0.5 (15.98)	-0.4 (30.83)	2.1 (51.51)
DR-2 (CMSE-OLS)	-0.2 (1.58)	-0.4 (7.95)	-0.6 (16.00)	-0.5 (30.87)	2.0 (51.64)
DR-3 (CMSE-OLS)	-0.2 (1.58)	-0.4 (7.95)	-0.6 (16.00)	-0.5 (30.89)	2.0 (51.76)
DR-4 (CMSE-OLS)	-0.2 (1.58)	-0.4 (7.95)	-0.6 (16.01)	-0.6 (30.92)	2.1 (51.80)
DR-5 (CMSE-OLS)	-0.2 (1.58)	-0.4 (7.95)	-0.6 (16.02)	-0.6 (30.93)	2.0 (51.85)
BEST (CMSE-OLS)	0.5 (1.61)	1.5 (8.15)	1.4 (16.10)	1.4 (30.92)	3.9 (51.76)
DR-1 (CMSE-Oracle)	0.3 (1.55)	0.4 (7.39)	-0.0 (14.42)	0.7 (26.00)	2.3 (40.68)
DR-2 (CMSE-Oracle)	-0.1 (1.55)	-0.4 (7.43)	-0.7 (14.46)	0.1 (26.08)	1.9 (40.78)
DR-3 (CMSE-Oracle)	-0.2 (1.55)	-0.4 (7.44)	-0.7 (14.48)	0.1 (26.15)	1.9 (40.89)
DR-4 (CMSE-Oracle)	-0.2 (1.55)	-0.4 (7.46)	-0.7 (14.51)	0.1 (26.23)	1.9 (41.02)
DR-5 (CMSE-Oracle)	-0.2 (1.55)	-0.4 (7.48)	-0.7 (14.55)	0.0 (26.31)	1.9 (41.17)
BEST (CMSE-Oracle)	0.3 (1.55)	0.6 (7.39)	1.0 (14.41)	2.5 (25.97)	6.5 (40.71)

Table 4.5: Comparison of different estimators of CMSE under medium sparsity (40% zero covariates) at corr = 0.3, showing % Relative Bias and MSE in brackets.

Estimator	$P/E(n_r)$				
	$\frac{10}{100}$	$\frac{20}{100}$	$\frac{30}{100}$	$\frac{40}{100}$	$\frac{50}{100}$
Ridge (CMSE)	0.4 (0.29)	1.3 (1.14)	2.5 (2.43)	4.8 (4.20)	5.4 (5.98)
DR-1 (CMSE-R)	0.1 (0.29)	0.9 (1.10)	1.6 (2.25)	3.3 (3.78)	2.6 (5.44)
DR-2 (CMSE-R)	0.0 (0.29)	0.3 (1.09)	0.6 (2.26)	1.9 (3.84)	1.3 (5.66)
DR-3 (CMSE-R)	0.0 (0.29)	0.0 (1.10)	0.2 (2.31)	1.4 (3.95)	0.9 (5.89)
DR-4 (CMSE-R)	0.0 (0.29)	-0.1 (1.12)	0.1 (2.36)	1.1 (4.05)	0.6 (6.10)
DR-5 (CMSE-R)	0.0 (0.29)	-0.1 (1.12)	0.0 (2.40)	0.9 (4.14)	0.5 (6.27)
BEST (CMSE-R)	0.3 (0.29)	1.2 (1.13)	2.3 (2.35)	4.3 (4.00)	4.4 (5.82)
DR-1 (CMSE-OLS)	0.3 (0.29)	0.7 (1.11)	0.9 (2.35)	1.8 (4.13)	1.0 (6.53)
DR-2 (CMSE-OLS)	0.2 (0.28)	0.3 (1.09)	0.3 (2.35)	1.2 (4.12)	0.5 (6.58)
DR-3 (CMSE-OLS)	0.1 (0.28)	0.1 (1.10)	0.2 (2.37)	1.0 (4.16)	0.4 (6.65)
DR-4 (CMSE-OLS)	0.0 (0.28)	0.0 (1.10)	0.1 (2.38)	0.9 (4.20)	0.3 (6.72)
DR-5 (CMSE-OLS)	0.0 (0.28)	0.0 (1.11)	0.0 (2.40)	0.8 (4.23)	0.2 (6.78)
BEST (CMSE-OLS)	0.3 (0.29)	0.9 (1.13)	1.7 (2.48)	3.1 (4.39)	2.8 (6.79)
DR-1 (CMSE-Oracle)	0.3 (0.28)	1.0 (1.07)	1.6 (2.19)	3.3 (3.74)	2.7 (5.41)
DR-2 (CMSE-Oracle)	0.2 (0.28)	0.5 (1.07)	0.6 (2.24)	1.9 (3.82)	1.3 (5.64)
DR-3 (CMSE-Oracle)	0.1 (0.28)	0.2 (1.09)	0.3 (2.29)	1.4 (3.93)	0.9 (5.88)
DR-4 (CMSE-Oracle)	0.0 (0.28)	0.1 (1.10)	0.1 (2.33)	1.1 (4.03)	0.7 (6.08)
DR-5 (CMSE-Oracle)	0.0 (0.28)	0.0 (1.11)	0.0 (2.37)	0.9 (4.11)	0.5 (6.25)
BEST (CMSE-Oracle)	0.3 (0.28)	0.9 (1.06)	1.7 (2.20)	3.3 (3.74)	3.2 (5.43)

Table 4.6: Comparison of different estimators of CMSE under medium sparsity (40% zero covariates) at corr = 0.95, showing % Relative Bias and MSE in brackets.

Estimator	$P/E(n_r)$				
	$\frac{10}{100}$	$\frac{20}{100}$	$\frac{30}{100}$	$\frac{40}{100}$	$\frac{50}{100}$
Ridge (CMSE)	0.6 (0.64)	1.1 (3.00)	1.3 (6.35)	2.0 (10.17)	1.2 (15.33)
DR-1 (CMSE-R)	0.2 (0.64)	0.4 (2.99)	0.3 (6.34)	0.8 (10.17)	0.1 (15.36)
DR-2 (CMSE-R)	0.2 (0.64)	0.2 (2.99)	0.2 (6.33)	0.8 (10.16)	0.0 (15.33)
DR-3 (CMSE-R)	0.2 (0.64)	0.2 (2.99)	0.2 (6.32)	0.8 (10.15)	0.0 (15.32)
DR-4 (CMSE-R)	0.2 (0.64)	0.2 (2.99)	0.2 (6.31)	0.8 (10.16)	0.0 (15.34)
DR-5 (CMSE-R)	0.2 (0.64)	0.2 (2.99)	0.2 (6.33)	0.7 (10.18)	0.0 (15.39)
BEST (CMSE-R)	0.5 (0.64)	1.0 (3.00)	1.2 (6.35)	1.8 (10.19)	1.0 (15.34)
DR-1 (CMSE-OLS)	0.3 (0.65)	0.2 (3.14)	0.4 (6.94)	0.6 (11.89)	0.0 (19.75)
DR-2 (CMSE-OLS)	0.2 (0.65)	0.1 (3.14)	0.3 (6.95)	0.6 (11.89)	0.0 (19.81)
DR-3 (CMSE-OLS)	0.2 (0.65)	0.1 (3.14)	0.3 (6.96)	0.6 (11.91)	0.0 (19.84)
DR-4 (CMSE-OLS)	0.2 (0.65)	0.1 (3.14)	0.3 (6.96)	0.6 (11.92)	0.0 (19.86)
DR-5 (CMSE-OLS)	0.2 (0.65)	0.1 (3.14)	0.3 (6.96)	0.6 (11.92)	0.0 (19.87)
BEST (CMSE-OLS)	0.6 (0.66)	1.2 (3.21)	1.6 (7.00)	1.7 (11.96)	1.1 (19.69)
DR-1 (CMSE-Oracle)	0.5 (0.63)	0.7 (2.95)	0.6 (6.24)	1.2 (10.08)	0.3 (15.17)
DR-2 (CMSE-Oracle)	0.2 (0.63)	0.3 (2.96)	0.2 (6.26)	0.8 (10.10)	0.0 (15.22)
DR-3 (CMSE-Oracle)	0.2 (0.63)	0.2 (2.97)	0.2 (6.27)	0.8 (10.12)	0.0 (15.27)
DR-4 (CMSE-Oracle)	0.1 (0.64)	0.2 (2.97)	0.2 (6.29)	0.8 (10.14)	0.0 (15.32)
DR-5 (CMSE-Oracle)	0.1 (0.64)	0.2 (2.97)	0.2 (6.30)	0.7 (10.17)	0.0 (15.38)
BEST (CMSE-Oracle)	0.5 (0.63)	0.9 (2.95)	1.7 (6.24)	2.3 (10.05)	2.6 (15.11)

The goal is to evaluate the relative performance of these methods in terms of Monte Carlo percent relative bias and mean squared error under different simulation settings.

The simulation results are summarized in Tables (4.7)-(4.12). The simulation results show several clear patterns across the different settings. Overall, the debiased ridge estimators with λ selected using the CMSE criterion perform well in all scenarios. When the debiasing order m is fixed, the estimators DR-1, DR-2, and DR-3 usually have very small bias and competitive mean squared error when compared to other estimators. For example, in the non-sparse setting with $\rho = 0.3$ and $p = 50$, DR-5 (CMSE) has a relative bias of 0.8 and an MSE of 16.27, whereas Ridge (CV) has a much larger bias of 34.3 and a larger MSE of 24.94. This shows that the debiasing step can substantially improve the ridge estimator.

When Ridge (CMSE) is compared with the debiased ridge estimators based on CMSE, the differences are usually present but not very large. In many cases, the debiased versions reduce the bias somewhat and may also slightly reduce the MSE, but the gain is not dramatic. For instance, in the non-sparse setting with $\rho = 0.3$ and $p = 40$, Ridge (CMSE) has an MSE of 10.21, while DR-3 (CMSE) has an MSE of 9.69. So the CMSE-based debiased ridge estimators are generally a bit better than ridge, but not by a large margin. This suggests that much of the gain already comes from choosing λ using CMSE instead of cross-validation.

The estimator BEST (CMSE), which jointly selects both m and λ, also performs well overall. However, it is not systematically better than fixing a small value of m. In many settings, DR-2 (CMSE) or DR-3 (CMSE) performs about as well as BEST (CMSE), and sometimes slightly better. This suggests that, in practice, fixing a small order such as $m = 2$ or $m = 3$ may already be enough.

The comparison with the LASSO-based methods is also interesting. Across all scenarios, Post-LASSO performs much better than Adaptive LASSO. Adaptive LASSO often has a fairly large bias, especially when p increases. For example, in the non-sparse case with $\rho = 0.3$ and $p = 50$, Adaptive LASSO has a relative bias of 29.8 and an MSE of 24.74, whereas Post-LASSO has a relative bias of -3.2 and an MSE of 15.28. So Post-LASSO is clearly preferable to Adaptive LASSO in these simulations.

In the non-sparse settings, the ridge-based methods tend to perform better than the LASSO-based methods. This is not surprising, since when all coefficients are nonzero, methods based on shrinkage are often more suitable than methods based on variable selection. For example, in the case $\rho = 0.95$ and $p = 50$, DR-2 (CMSE) has an MSE of 40.76, while Post-LASSO has an MSE of 44.14 and Adaptive LASSO has an MSE of 47.71. In this setting, the debiased ridge estimators are clearly better.

When sparsity is introduced, Post-LASSO becomes more competitive. In the medium sparsity setting (40% zero coefficients), the ridge-based methods still perform very well, but the

gap with Post-LASSO becomes smaller. For example, when $\rho = 0.3$ and $P = 50$, Post-LASSO has an MSE of 5.96, while DR-5 (CMSE) has an MSE of 6.27. These values are very close, which suggests that Post-LASSO can be quite effective once sparsity is present.

This effect is even more visible in the high sparsity setting (60% zero coefficients). In this case, Post-LASSO performs about as well as the best ridge-based methods, and sometimes slightly better. For instance, when $\rho = 0.3$ and $p = 50$, Post-LASSO has an MSE of 3.22, while DR-5 (CMSE) has an MSE of 3.27. So in strongly sparse settings, Post-LASSO becomes a very competitive alternative.

Finally, the effect of the correlation level is also important. When the correlation is high, OLS becomes much less stable. For example, in the non-sparse case with $\rho = 0.95$ and $p = 50$, OLS has an MSE of 63.80, which is much larger than the corresponding values for ridge and debiased ridge estimators.

Table 4.7: Comparison of imputation methods with no sparsity at corr = 0.3, showing % Relative Bias and MSE in brackets.

Estimator	$P/E(n_r)$				
	$\frac{10}{100}$	$\frac{20}{100}$	$\frac{30}{100}$	$\frac{40}{100}$	$\frac{50}{100}$
Full	0.1 (0.34)	0.1 (1.27)	-0.2 (2.41)	0.1 (4.02)	-0.0 (5.88)
OLS	0.1 (0.55)	-0.1 (2.57)	-0.1 (5.72)	0.8 (11.31)	-0.5 (20.07)
Adaptive Lasso (CV)	1.9 (0.61)	8.6 (3.43)	16.3 (8.34)	23.8 (15.79)	29.8 (24.74)
Post-LASSO	0.3 (0.55)	0.2 (2.54)	-0.1 (5.35)	-0.9 (9.77)	-3.2 (15.28)
Ridge (CV)	3.0 (0.63)	10.1 (3.46)	18.2 (8.22)	26.6 (15.43)	34.3 (24.94)
DR-1 (CV)	0.5 (0.55)	1.8 (2.49)	4.1 (5.20)	6.9 (9.40)	9.1 (14.57)
DR-2 (CV)	0.2 (0.54)	0.5 (2.49)	1.5 (5.15)	3.2 (9.28)	4.2 (14.29)
DR-3 (CV)	0.1 (0.54)	0.1 (2.51)	0.8 (5.24)	2.3 (9.47)	2.8 (14.61)
DR-4 (CV)	0.1 (0.54)	0.0 (2.53)	0.6 (5.32)	1.9 (9.67)	2.1 (14.98)
DR-5 (CV)	0.1 (0.55)	-0.0 (2.54)	0.4 (5.38)	1.6 (9.85)	1.7 (15.32)
Ridge (CMSE)	0.7 (0.55)	2.2 (2.60)	4.3 (5.57)	7.3 (10.21)	9.4 (15.80)
DR-1 (CMSE)	0.5 (0.55)	1.6 (2.50)	3.0 (5.11)	4.4 (9.17)	4.9 (14.11)
DR-2 (CMSE)	0.2 (0.54)	0.6 (2.47)	1.2 (5.14)	2.4 (9.40)	2.4 (14.68)
DR-3 (CMSE)	0.1 (0.54)	0.2 (2.50)	0.7 (5.26)	1.8 (9.69)	1.6 (15.29)
DR-4 (CMSE)	0.1 (0.55)	0.0 (2.53)	0.4 (5.36)	1.5 (9.95)	1.1 (15.82)
DR-5 (CMSE)	0.1 (0.55)	-0.0 (2.54)	0.3 (5.44)	1.3 (10.16)	0.8 (16.27)
BEST (CMSE)	0.5 (0.55)	1.9 (2.54)	3.6 (5.30)	6.1 (9.71)	7.4 (15.10)

Table 4.8: Comparison of imputation methods with no sparsity at corr = 0.95, showing % Relative Bias and MSE in brackets.

Estimator	$P/E(n_r)$				
	$\frac{10}{100}$	$\frac{20}{100}$	$\frac{30}{100}$	$\frac{40}{100}$	$\frac{50}{100}$
Full	0.0 (0.82)	0.1 (3.63)	-0.5 (6.86)	0.1 (11.66)	-0.7 (18.20)
OLS	-0.1 (1.55)	-0.2 (8.25)	-0.5 (17.90)	-0.3 (36.00)	-0.9 (63.80)
Adaptive Lasso (CV)	3.1 (1.65)	8.8 (8.61)	12.3 (17.33)	16.2 (30.67)	19.2 (47.71)
Post-LASSO	-0.3 (1.52)	-1.6 (7.69)	-3.5 (15.50)	-5.4 (28.20)	-7.8 (44.14)
Ridge (CV)	2.2 (1.65)	5.7 (7.81)	8.0 (15.51)	10.5 (27.56)	12.3 (42.08)
DR-1 (CV)	0.1 (1.50)	0.3 (7.51)	0.3 (14.85)	0.4 (26.63)	-0.1 (40.84)
DR-2 (CV)	0.0 (1.50)	0.0 (7.56)	-0.2 (14.99)	-0.1 (26.92)	-0.8 (41.29)
DR-3 (CV)	-0.0 (1.51)	-0.0 (7.61)	-0.2 (15.13)	-0.1 (27.21)	-0.9 (41.74)
DR-4 (CV)	-0.0 (1.51)	-0.0 (7.65)	-0.3 (15.25)	-0.1 (27.48)	-0.9 (42.16)
DR-5 (CV)	-0.0 (1.52)	-0.0 (7.69)	-0.3 (15.36)	-0.1 (27.73)	-0.9 (42.56)
Ridge (CMSE)	0.6 (1.50)	1.5 (7.51)	1.6 (14.71)	1.8 (26.61)	1.2 (40.71)
DR-1 (CMSE)	0.1 (1.50)	0.3 (7.49)	-0.0 (14.73)	-0.0 (26.64)	-0.7 (40.86)
DR-2 (CMSE)	-0.0 (1.50)	0.0 (7.49)	-0.2 (14.68)	-0.1 (26.55)	-0.7 (40.76)
DR-3 (CMSE)	-0.0 (1.50)	0.0 (7.49)	-0.2 (14.68)	-0.1 (26.52)	-0.8 (40.73)
DR-4 (CMSE)	-0.0 (1.50)	0.0 (7.50)	-0.2 (14.69)	-0.1 (26.56)	-0.8 (40.79)
DR-5 (CMSE)	-0.0 (1.50)	0.0 (7.50)	-0.3 (14.71)	-0.2 (26.64)	-0.8 (40.92)
BEST (CMSE)	0.5 (1.50)	1.3 (7.51)	1.4 (14.72)	1.6 (26.62)	0.9 (40.81)

Table 4.9: Comparison of imputation methods with medium sparsity (40% zero covariates) at corr = 0.3, showing % Relative Bias and MSE in brackets.

Estimator	$P/E(n_r)$				
	$\frac{10}{100}$	$\frac{20}{100}$	$\frac{30}{100}$	$\frac{40}{100}$	$\frac{50}{100}$
Full	-0.1 (0.17)	0.0 (0.56)	-0.1 (1.03)	0.5 (1.66)	-0.2 (2.42)
OLS	-0.1 (0.29)	-0.2 (1.13)	-0.2 (2.51)	0.4 (4.61)	-0.3 (7.74)
Adaptive Lasso (CV)	0.9 (0.30)	4.4 (1.36)	8.4 (3.26)	13.6 (6.03)	16.2 (8.74)
Post-LASSO	0.0 (0.28)	-0.4 (1.10)	-1.1 (2.36)	-0.9 (4.00)	-2.9 (5.96)
Ridge (CV)	1.2 (0.30)	5.6 (1.41)	10.1 (3.34)	16.1 (6.19)	19.0 (8.72)
DR-1 (CV)	0.1 (0.28)	0.9 (1.09)	2.0 (2.29)	4.5 (3.90)	4.6 (5.50)
DR-2 (CV)	0.1 (0.29)	0.2 (1.10)	0.6 (2.28)	2.3 (3.83)	2.1 (5.50)
DR-3 (CV)	-0.0 (0.29)	-0.0 (1.11)	0.3 (2.33)	1.7 (3.90)	1.4 (5.64)
DR-4 (CV)	-0.0 (0.29)	-0.1 (1.11)	0.1 (2.36)	1.4 (3.97)	1.1 (5.79)
DR-5 (CV)	-0.0 (0.29)	-0.1 (1.12)	0.0 (2.39)	1.2 (4.04)	0.9 (5.93)
Ridge (CMSE)	0.4 (0.29)	1.3 (1.14)	2.5 (2.43)	4.8 (4.20)	5.4 (5.98)
DR-1 (CMSE)	0.1 (0.29)	0.9 (1.10)	1.6 (2.25)	3.3 (3.78)	2.6 (5.44)
DR-2 (CMSE)	0.0 (0.29)	0.3 (1.09)	0.6 (2.26)	1.9 (3.84)	1.3 (5.66)
DR-3 (CMSE)	-0.0 (0.29)	0.0 (1.10)	0.2 (2.31)	1.4 (3.95)	0.9 (5.89)
DR-4 (CMSE)	-0.0 (0.29)	-0.1 (1.12)	0.1 (2.36)	1.1 (4.05)	0.6 (6.09)
DR-5 (CMSE)	-0.0 (0.29)	-0.1 (1.12)	-0.0 (2.40)	0.9 (4.14)	0.5 (6.27)
BEST (CMSE)	0.3 (0.29)	1.2 (1.23)	2.3 (2.35)	4.3 (4.00)	4.4 (5.82)

Table 4.10: Comparison of imputation methods with medium sparsity (40% zero covariates) at corr = 0.95, showing % Relative Bias and MSE in brackets.

Estimator	$P/E(n_r)$				
	$\frac{10}{100}$	$\frac{20}{100}$	$\frac{30}{100}$	$\frac{40}{100}$	$\frac{50}{100}$
Full	0.1 (0.35)	0.0 (1.41)	0.0 (2.82)	0.5 (4.48)	0.4 (6.82)
OLS	0.2 (0.66)	-0.0 (3.33)	0.4 (7.67)	0.3 (13.68)	-0.3 (23.59)
Adaptive Lasso (CV)	2.1 (0.65)	5.5 (3.47)	7.8 (6.82)	10.6 (12.07)	12.0 (17.96)
Post-LASSO	-0.0 (0.65)	-0.9 (3.05)	-2.0 (6.64)	-2.6 (10.83)	-4.1 (16.40)
Ridge (CV)	1.5 (0.66)	3.7 (3.15)	5.2 (6.63)	7.3 (10.70)	7.8 (15.97)
DR-1 (CV)	0.2 (0.64)	0.4 (3.00)	0.4 (6.36)	1.2 (10.20)	0.4 (15.36)
DR-2 (CV)	0.2 (0.64)	0.2 (3.02)	0.2 (6.42)	0.8 (10.28)	0.0 (15.50)
DR-3 (CV)	0.2 (0.64)	0.1 (3.04)	0.2 (6.47)	0.7 (10.37)	-0.0 (15.65)
DR-4 (CV)	0.2 (0.65)	0.1 (3.06)	0.2 (6.53)	0.7 (10.45)	-0.0 (15.81)
DR-5 (CV)	0.2 (0.65)	0.1 (3.07)	0.2 (6.57)	0.7 (10.53)	-0.0 (15.96)
Ridge (CMSE)	0.6 (0.64)	1.1 (3.00)	1.3 (6.35)	1.9 (10.17)	1.2 (15.33)
DR-1 (CMSE)	0.2 (0.64)	0.4 (2.99)	0.3 (6.34)	0.8 (10.17)	0.1 (15.36)
DR-2 (CMSE)	0.2 (0.64)	0.2 (2.99)	0.2 (6.33)	0.8 (10.16)	0.0 (15.33)
DR-3 (CMSE)	0.2 (0.64)	0.2 (2.99)	0.2 (6.32)	0.8 (10.15)	0.0 (15.32)
DR-4 (CMSE)	0.2 (0.64)	0.2 (2.99)	0.2 (6.31)	0.8 (10.16)	0.0 (15.34)
DR-5 (CMSE)	0.2 (0.64)	0.2 (3.00)	0.2 (6.33)	0.7 (10.18)	0.0 (15.39)
BEST (CMSE)	0.5 (0.64)	1.0 (3.00)	1.2 (6.35)	1.8 (10.19)	1.0 (15.34)

Table 4.11: Comparison of imputation methods with high sparsity (60% zero covariates) at corr = 0.3, showing % Relative Bias and MSE in brackets.

Estimator	$P/E(n_r)$				
	$\frac{10}{100}$	$\frac{20}{100}$	$\frac{30}{100}$	$\frac{40}{100}$	$\frac{50}{100}$
Full	0.1 (0.10)	0.1 (0.24)	0.0 (0.53)	0.1 (0.85)	-0.0 (1.22)
OLS	0.0 (0.17)	0.1 (0.47)	-0.0 (1.29)	-0.1 (2.42)	-0.4 (3.96)
Adaptive Lasso (CV)	0.7 (0.17)	2.4 (0.54)	5.0 (1.55)	7.7 (2.89)	9.8 (4.21)
Post-LASSO	0.0 (0.17)	-0.2 (0.46)	-1.0 (1.23)	-1.6 (2.14)	-2.5 (3.22)
Ridge (CV)	0.6 (0.17)	3.1 (0.53)	6.5 (1.62)	9.6 (2.96)	12.3 (4.37)
DR-1 (CV)	0.1 (0.17)	0.6 (0.46)	1.4 (1.20)	2.2 (2.04)	3.0 (2.97)
DR-2 (CV)	0.0 (0.17)	0.3 (0.46)	0.6 (1.20)	1.0 (2.04)	1.3 (2.97)
DR-3 (CV)	0.0 (0.17)	0.2 (0.47)	0.3 (1.22)	0.6 (2.08)	0.8 (3.03)
DR-4 (CV)	0.0 (0.17)	0.1 (0.47)	0.2 (1.23)	0.5 (2.12)	0.6 (3.10)
DR-5 (CV)	0.0 (0.17)	0.1 (0.47)	0.2 (1.24)	0.4 (2.15)	0.4 (3.16)
Ridge (CMSE)	0.3 (0.17)	1.0 (0.48)	1.9 (1.26)	2.8 (2.18)	3.7 (3.17)
DR-1 (CMSE)	0.0 (0.17)	0.6 (0.46)	1.3 (1.18)	1.6 (1.99)	1.8 (2.90)
DR-2 (CMSE)	0.0 (0.17)	0.3 (0.46)	0.6 (1.18)	0.8 (2.03)	0.8 (2.98)
DR-3 (CMSE)	0.0 (0.17)	0.2 (0.47)	0.3 (1.20)	0.5 (2.09)	0.5 (3.09)
DR-4 (CMSE)	0.0 (0.17)	0.1 (0.47)	0.2 (1.23)	0.3 (2.14)	0.3 (3.18)
DR-5 (CMSE)	0.0 (0.17)	0.1 (0.48)	0.1 (1.24)	0.2 (2.19)	0.1 (3.27)
BEST (CMSE)	0.3 (0.17)	1.0 (0.48)	1.9 (1.23)	2.6 (2.11)	3.2 (3.07)

Table 4.12: Comparison of imputation methods with high sparsity (60% zero covariates) at corr = 0.95, showing % Relative Bias and MSE in brackets.

Estimator	$P/E(n_r)$				
	$\frac{10}{100}$	$\frac{20}{100}$	$\frac{30}{100}$	$\frac{40}{100}$	$\frac{50}{100}$
Full	0.1 (0.19)	0.1 (0.56)	0.2 (1.42)	0.5 (2.29)	-0.2 (3.38)
OLS	0.2 (0.37)	-0.0 (1.27)	0.1 (3.66)	0.6 (6.99)	0.4 (11.55)
Adaptive Lasso (CV)	1.5 (0.36)	3.3 (1.18)	5.5 (3.22)	7.5 (6.31)	8.5 (8.93)
Post-LASSO	0.0 (0.36)	-0.6 (1.18)	-1.4 (3.20)	-1.5 (5.97)	-2.7 (8.14)
Ridge (CV)	1.1 (0.39)	2.2 (1.20)	3.7 (3.24)	5.2 (5.59)	5.6 (7.89)
DR-1 (CV)	0.2 (0.36)	0.2 (1.16)	0.4 (3.07)	1.0 (5.28)	0.5 (7.55)
DR-2 (CV)	0.2 (0.36)	0.1 (1.16)	0.2 (3.09)	0.8 (5.33)	0.3 (7.63)
DR-3 (CV)	0.2 (0.36)	0.1 (1.17)	0.2 (3.11)	0.8 (5.38)	0.2 (7.70)
DR-4 (CV)	0.2 (0.36)	0.1 (1.18)	0.2 (3.14)	0.7 (5.42)	0.2 (7.76)
DR-5 (CV)	0.2 (0.36)	0.0 (1.18)	0.2 (3.16)	0.7 (5.47)	0.2 (7.85)
Ridge (CMSE)	0.5 (0.36)	0.6 (1.15)	1.0 (3.06)	1.6 (5.25)	1.1 (7.56)
DR-1 (CMSE)	0.2 (0.36)	0.2 (1.15)	0.3 (3.04)	0.9 (5.24)	0.3 (7.56)
DR-2 (CMSE)	0.2 (0.35)	0.1 (1.15)	0.2 (3.07)	0.8 (5.23)	0.3 (7.54)
DR-3 (CMSE)	0.2 (0.36)	0.1 (1.15)	0.2 (3.06)	0.8 (5.23)	0.3 (7.52)
DR-4 (CMSE)	0.2 (0.36)	0.1 (1.15)	0.2 (3.06)	0.8 (5.24)	0.3 (7.52)
DR-5 (CMSE)	0.2 (0.36)	0.1 (1.15)	0.2 (3.06)	0.8 (5.25)	0.3 (7.55)
BEST (CMSE)	0.4 (0.35)	0.5 (1.15)	0.9 (3.06)	1.5 (5.25)	1.0 (7.57)

Chapter 5

Mean Square Error Estimation

5.1 Introduction

In Chapter 4, we discussed the behavior of point estimators when ridge regression and debiased ridge regression methods are used for imputation. In particular, we studied the bias and efficiency of the resulting imputed estimators. An equally important aspect in practice is the assessment of their variance. National statistical offices typically disseminate not only point estimates but also measures of uncertainty. In most cases, uncertainty is reported through an estimated variance or, more commonly, through an estimated coefficient of variation (CV). For an estimator $\widehat{\theta}$ of a finite population parameter θ, the estimated coefficient of variation is defined as $\widehat{\mathrm{CV}}(\widehat{\theta}) = \sqrt{\widehat{\mathbb{V}}(\widehat{\theta})}/\widehat{\theta}$, where $\widehat{\mathbb{V}}(\widehat{\theta})$ is an estimate of the variance of $\widehat{\theta}$. When imputation is used to handle missing data, variance estimation becomes more challenging. Moreover, in the context of ridge and debiased ridge imputation, focusing only on variance is not sufficient, since these estimators may be biased. As a result, it is more appropriate to consider the mean squared error (MSE), which accounts for both variance and bias. In this chapter we focus on the estimation of the mean squared error (MSE) of the imputed estimator obtained based on ridge and debiased ridge imputation. We consider two approaches to MSE estimation. The first approach is based on the variance estimation framework proposed by Sarndal [1992], which provides a decomposition of the total variance into components due to sampling and nonresponse. The second approach relies on a bootstrap method, more specifically a pseudo-population bootstrap, designed to replicate the combined effects of the sampling design, the nonresponse mechanism, and the imputation procedure. These two approaches are studied and compared in order to assess their performance for variance and MSE estimation when ridge-type imputation methods are used.

5.2 Estimation of the MSE: Särndal's method

Within the *mpq* inferential framework, the total error of an imputed estimator can be viewed as the result of two main sources: sampling and nonresponse. First, a sample S is selected from the finite population U according to the sampling design p. Then, among the selected units, only a subset S_r responds, while the remaining units S_m are nonrespondents according to the nonresponse mechanism q. This two-stage process can be summarized as

$$U \xrightarrow{p} S \xrightarrow{q} (S_r, S_m).$$

As a result, the error of the imputed estimator reflects both the variability due to sampling and the additional variability due to nonresponse and imputation. In this context, it is natural to evaluate the performance of the imputed estimator of the population total, $\hat{t}_I$, through its mean squared error (MSE) under the joint distribution induced by the imputation model m, the sampling design p, and the nonresponse mechanism q.

When the imputed estimator $\hat{t}_I$ may be biased, its performance under the (m, p, q) framework must be evaluated through its mean squared error rather than only its variance. The mean squared error can be written as

$$\begin{aligned} \mathrm{MSE}_{mpq}(\hat{t}_I) &= \mathbb{E}_m \mathbb{E}_p \mathbb{E}_q \big[(\hat{t}_I - t_y)^2\big] \\ &= \mathbb{V}_{mpq}(\hat{t}_I) + \Big\{\mathrm{Bias}_{mpq}(\hat{t}_I)\Big\}^2, \end{aligned} \tag{5.2.1}$$

where

$$\mathrm{Bias}_{mpq}(\hat{t}_I) = \mathbb{E}_m \mathbb{E}_p \mathbb{E}_q (\hat{t}_I - t_y)$$

denotes the total bias under the joint (m, p, q) distribution. When the imputation procedure is unbiased, this bias term vanishes and the mean squared error reduces to the total variance. Otherwise, the squared bias must be taken into account.

To derive an expression of the variance, we start with the decomposition of the total error Sarndal [1992]:

$$\hat{t}_I - t_y = (\hat{t}_{\mathrm{HT}} - t_y) + (\hat{t}_I - \hat{t}_{\mathrm{HT}}).$$

Squaring both sides and taking the expectation under the joint (m, p, q) distribution leads to the decomposition

$$\mathbb{V}_{mpq}(\hat{t}_I) = \mathbb{V}_{\mathrm{sam}} + \mathbb{V}_{\mathrm{nr}} + 2\,\mathbb{V}_{\mathrm{mix}}. \tag{5.2.2}$$

The term $\mathbb{V}_{\mathrm{sam}} = \mathbb{E}_m \mathbb{V}_p(\hat{t}_{\mathrm{HT}})$ denotes the sampling variance. It corresponds to the variability induced by the sampling design and represents the variance of the Horvitz–Thompson estimator that would be observed if complete data were available. This component is not affected by the presence of imputation bias. The term

$$\mathbb{V}_{\mathrm{nr}} = \mathbb{E}_p \mathbb{E}_q \mathbb{V}_m(\hat{t}_I - \hat{t}_{\mathrm{HT}})$$

denotes the additional variability due to nonresponse and imputation. It reflects the uncertainty introduced by the imputation model and by the fact that only a subset of sampled units provides observed values. Finally,

$$\mathbb{V}_{\text{mix}} = \mathbb{E}_p \mathbb{E}_q \text{Cov}_m\big(\hat{t}_{\text{HT}} - t_y,\, \hat{t}_I - \hat{t}_{\text{HT}}\big)$$

is a mixed component that captures the interaction between sampling error and nonresponse error.

Combining these results, the mean squared error of the imputed estimator can be expressed as

$$\text{MSE}_{mpq}(\hat{t}_I) = \mathbb{V}_{\text{sam}} + \mathbb{V}_{\text{nr}} + 2\,\mathbb{V}_{\text{mix}} + \Big\{\text{Bias}_{mpq}(\hat{t}_I)\Big\}^2. \tag{5.2.3}$$

5.2.1 Estimation of $\mathbb{V}_{\text{sam}}$

As a starting point for estimating the sampling variance component $\mathbb{V}_{\text{sam}}$, one may consider a naive variance estimator that treats the imputed values as if they were true observed values. This approach consists of applying a standard complete-data variance estimation procedure to the "completed" variable $\tilde{y}$, defined for each sampled unit $i \in S$ as

$$\tilde{y}_i = r_i y_i + (1 - r_i) y_i^*,$$

where y_i^* denotes the imputed value. By applying the usual Horvitz–Thompson variance formula to the completed values $\tilde{y}_i$, the naive variance estimator is obtained as

$$\widehat{\mathbb{V}}_{\text{naive}} = \sum_{i \in S} \sum_{j \in S} \frac{\pi_{ij} - \pi_i \pi_j}{\pi_{ij}} \frac{\tilde{y}_i}{\pi_i} \frac{\tilde{y}_j}{\pi_j}. \tag{5.2.4}$$

This estimator is generally biased for the true sampling variance $\mathbb{V}_{\text{sam}}$, typically leading to underestimation because the imputed values are treated as if they were observed. Following Sarndal [1992], the idea is therefore to evaluate and estimate this bias, and then to add a correction term to the naive estimator.

Let $\widehat{\mathbb{V}}_p(\widehat{t}_{y,HT})$ denote the usual Horvitz–Thompson variance estimator computed from the complete (but unobserved) data y_i; see Expression (3.2.5). The bias of the naive estimator can be expressed through the average difference

$$\mathbb{V}_{\text{diff}} = \mathbb{E}_m\Big(\widehat{\mathbb{V}}_p(\widehat{t}_{y,HT}) - \widehat{\mathbb{V}}_{\text{naive}}\Big),$$

which represents the expected gap, under the imputation model, between the complete-data variance estimator and the naive estimator. In practice, direct evaluation of this expectation may be difficult for complex imputation procedures. Beaumont and Bocci [2009] proposed a

conditional approach in which expectations are taken conditionally on the observed respondent values $\mathbf{y}_r$. Since $\widehat{\mathbb{V}}_{\text{naive}}$ depends only on the observed y_i's, it can be treated as fixed under the conditional model expectation. This leads to

$$\mathbb{V}_{\text{diff}} = \mathbb{E}_m\Big(\widehat{\mathbb{V}}_p(\widehat{t}_{y,HT}) \mid \mathbf{y}_r\Big) - \widehat{\mathbb{V}}_{\text{naive}}. \tag{5.2.5}$$

It can be shown (See Appendix A.3) that

$$\mathbb{V}_{\text{diff}} = \sigma^2 \sum_{i\in S} w_i^2 \,(1-\pi_i)\,(1-r_i),$$

In practice, the model error variance σ^2 is unknown and must be estimated from the respondent data. A natural unbiased estimator of σ^2 under the linear model is

$$\widehat{\sigma}^2 = \frac{1}{n_r - p} \sum_{i\in S_r} \Big(y_i - \mathbf{x}_i^\top \widehat{\boldsymbol{\beta}}_r\Big)^2.$$

Here, $\widehat{\boldsymbol{\beta}}_r$ denotes the ordinary least-squares estimator computed from the respondent data.

Substituting $\widehat{\sigma}^2$ into the expression of $\mathbb{V}_{\text{diff}}$ yields the estimator

$$\widehat{\mathbb{V}}_{\text{diff}} = \widehat{\sigma}^2 \sum_{i\in S} w_i^2 \,(1-\pi_i)\,(1-r_i).$$

The corrected estimator of the sampling variance component is therefore obtained by adding this estimated difference term to the naive variance estimator, that is,

$$\widehat{\mathbb{V}}_{\text{sam}} = \widehat{\mathbb{V}}_{\text{naive}} + \widehat{\mathbb{V}}_{\text{diff}}.$$

5.2.2 Estimation of $\mathbb{V}_{\text{nr}}$

Under the Särndal variance decomposition framework, the nonresponse variance component is given by

$$\mathbb{V}_{\text{nr}} = \mathbb{E}_q \mathbb{E}_p \mathbb{V}_m\big(\hat{t}_I - \hat{t}_{\text{HT}}\big)\,. \tag{5.2.6}$$

Under deterministic linear imputation based on weighted least squares, we have

$$\mathbb{V}_{\text{nr}} = \sigma^2\, \mathbb{E}_p \mathbb{E}_q \left\{ \sum_{i\in S} w_i^2 \,(r_i g_i - 1)^2 \right\}, \tag{5.2.7}$$

where

$$g_i = 1 + \left(\sum_{j\in S} w_j \mathbf{x}_j - \sum_{j\in S_r} w_j \mathbf{x}_j\right)^\top \left(\sum_{j\in S_r} w_j \mathbf{x}_j \mathbf{x}_j^\top\right)^{-1} \mathbf{x}_i. \tag{5.2.8}$$

The detailed proof is provided in Appendix A.4.

More generally, suppose that the imputation model remains linear in the response vector $\mathbf{y}_r$ but that the regression coefficients are estimated using a method other than weighted least squares. In ridge-type imputation procedures and related methods, the estimator of the regression coefficients can often be written in the form

$$\widehat{\boldsymbol{\beta}}_r = \mathbf{M}\left(\sum_{j\in S_r} w_j \mathbf{x}_j y_j\right), \tag{5.2.9}$$

where $\mathbf{M}$ is a matrix that depends only on the auxiliary variables and on tuning parameters of the method. In this setting, the imputed estimator of the population total remains linear in the observed responses and can be expressed as

$$\hat{t}_I = \sum_{i\in S_r} w_i g_i y_i,$$

with adjustment factors

$$g_i = 1 + \left(\sum_{j\in S} w_j \mathbf{x}_j - \sum_{j\in S_r} w_j \mathbf{x}_j\right)^\top \mathbf{M}\,\mathbf{x}_i. \tag{5.2.10}$$

For instance, for weighted ridge regression,

$$\widehat{\boldsymbol{\beta}}_r^{\text{ridge}} = \left(\sum_{j\in S_r} w_j \mathbf{x}_j \mathbf{x}_j^\top + \lambda \mathbf{I}\right)^{-1}\left(\sum_{j\in S_r} w_j \mathbf{x}_j y_j\right),$$

so that $\mathbf{M} = \mathbf{A}^{-1}$ with

$$\mathbf{A} = \sum_{j\in S_r} w_j \mathbf{x}_j \mathbf{x}_j^\top + \lambda \mathbf{I}.$$

The corresponding adjustment factor becomes

$$g_i = 1 + \left(\sum_{j\in S} w_j \mathbf{x}_j - \sum_{j\in S_r} w_j \mathbf{x}_j\right)^\top \mathbf{A}^{-1}\mathbf{x}_i. \tag{5.2.11}$$

For the first-order debiased ridge estimator, we have

$$\mathbf{M} = (\mathbf{I} + \lambda \mathbf{A}^{-1})\mathbf{A}^{-1}.$$

The corresponding adjustment factor is therefore

$$g_i = 1 + \left(\sum_{j\in S} w_j \mathbf{x}_j - \sum_{j\in S_r} w_j \mathbf{x}_j\right)^\top (\mathbf{I} + \lambda \mathbf{A}^{-1})\mathbf{A}^{-1}\mathbf{x}_i. \tag{5.2.12}$$

Higher-order debiased ridge estimators can be treated in the same way. As long as the resulting imputed estimator remains linear in the observed responses, it admits a representation of the form (5.2.9), and the corresponding adjustment factors g_i are obtained by substituting

the appropriate matrix $\mathbf{M}$ into (5.2.10).

An estimator of $\mathbb{V}_{\text{nr}}$ is obtained by replacing the unknown model variance σ^2 in (5.2.7) with a consistent estimator $\widehat{\sigma}^2$, yielding

$$\widehat{\mathbb{V}}_{\text{nr}} = \widehat{\sigma}^2 \left(\sum_{i \in S} w_i^2 (r_i g_i - 1)^2 \right), \tag{5.2.13}$$

where g_i takes the appropriate form depending on the chosen imputation method.

5.2.3 Estimation of $\mathbb{V}_{\text{mix}}$

The interaction component, which represents the covariance between the sampling error and the imputation error, is given by

$$\mathbb{V}_{\text{mix}} = \mathbb{E}_p \mathbb{E}_q \text{Cov}_m \left\{ \left(\hat{t}_{\text{HT}} - t_y \right), \left(\hat{t}_I - \hat{t}_{\text{HT}} \right) \right\}. \tag{5.2.14}$$

Under deterministic linear imputation, the covariance component defined in (5.2.14) can be expressed as

$$\mathbb{V}_{\text{mix}} = \sigma^2 \, \mathbb{E}_p \mathbb{E}_q \left(\sum_{i \in S} w_i^2 (1 - \pi_i)(r_i g_i - 1) \right), \tag{5.2.15}$$

where the adjustment factors g_i are defined as in (5.2.8). The detailed proof is provided in Appendix A.5.

As in the case of the nonresponse component $\mathbb{V}_{\text{nr}}$, the above expression extends directly to any deterministic linear imputation method for which the imputed estimator remains linear in the observed responses. When the regression coefficients are estimated using a general linear estimator of the form

$$\widehat{\boldsymbol{\beta}}_r = \mathbf{M} \left(\sum_{j \in S_r} w_j \mathbf{x}_j y_j \right),$$

the imputed estimator can be written as

$$\hat{t}_I = \sum_{i \in S_r} w_i g_i y_i,$$

with

$$g_i = 1 + \left(\sum_{j \in S} w_j \mathbf{x}_j - \sum_{j \in S_r} w_j \mathbf{x}_j \right)^{\top} \mathbf{M} \, \mathbf{x}_i,$$

so that the same expression (5.2.15) remains valid with the appropriate choice of g_i.

An estimator of $\mathbb{V}_{\text{mix}}$ is obtained by replacing the unknown model variance σ^2 in (5.2.15) with a consistent estimator $\widehat{\sigma}^2$, yielding

$$\widehat{\mathbb{V}}_{\text{mix}} = \widehat{\sigma}^2 \sum_{i \in S} w_i(w_i - 1)(r_i g_i - 1), \tag{5.2.16}$$

where g_i takes the appropriate form depending on the chosen imputation method.

5.2.4 Estimation of the mean square error

Recall that mean square error of the imputed estimator is given by

$$\text{MSE}_{mpq}(\hat{t}_I) = \mathbb{E}_m \mathbb{E}_p \mathbb{E}_q \left[(\hat{t}_I - t_y)^2 \right].$$

Then,

$$\begin{aligned} \text{MSE}_{mpq}(\hat{t}_I) &= \mathbb{V}_{mpq}(\hat{t}_I) + \left\{ \text{Bias}_{mpq}(\hat{t}_I) \right\}^2 \\ &= \mathbb{V}_{\text{sam}} + \mathbb{V}_{\text{nr}} + 2\,\mathbb{V}_{\text{mix}} + \left\{ \text{Bias}_{mpq}(\hat{t}_I) \right\}^2, \end{aligned} \tag{5.2.17}$$

An estimator of the mean squared error is obtained by plugging in the estimators of the variance components together with an estimator of the bias:

$$\widehat{\text{MSE}}_{mpq}(\hat{t}_I) = \widehat{\mathbb{V}}_{\text{sam}} + \widehat{\mathbb{V}}_{\text{nr}} + 2\,\widehat{\mathbb{V}}_{\text{mix}} + \left\{ \widehat{\text{Bias}}_{mpq}(\hat{t}_I) \right\}^2. \tag{5.2.18}$$

In Chapter 3, we derived closed-form expressions for the bias when the imputed values are obtained using the mth-order debiased ridge regression estimator:

$$\text{Bias}_{mpq}(\widehat{t}_I^{DR,(m)}) = -\lambda^{m+1} \mathbb{E}_{pq} \left[\left(\sum_{i \in S_m} w_i \mathbf{x}_i \right)^\top (\mathbf{A}_r + \lambda \mathbf{I}_p)^{-(m+1)} \right] \boldsymbol{\beta}, \tag{5.2.19}$$

where

$$\mathbf{A}_r = \sum_{j \in S_r} w_j \mathbf{x}_j \mathbf{x}_j^\top.$$

An mpq-unbiased estimator of the bias in (5.2.19) is obtained by replacing $\boldsymbol{\beta}$ with the least squares estimator $\widehat{\boldsymbol{\beta}}$, leading to

$$\widehat{\text{Bias}}_{mpq}(\widehat{t}_I^{DR,(m)}) = -\lambda^{m+1} \left(\sum_{i \in S_m} w_i \mathbf{x}_i \right)^\top (\mathbf{A}_r + \lambda \mathbf{I}_p)^{-(m+1)} \widehat{\boldsymbol{\beta}}. \tag{5.2.20}$$

We end this section by noting that all components entering the mean squared error estimator in (5.2.18) depend on the ridge penalty parameter λ and, when debiased ridge imputation is used, on the order of debiasing m. In practice, there are two approaches: either m is fixed and λ is selected by cross-validation or by minimizing an estimate of the conditional mean squared

error (CMSE) proposed in Chapter 3, or both λ and m are selected jointly by minimizing the estimated CMSE over a grid of candidate values. Although these choices are sample-dependent, the selected values of λ (or the pair (m, λ)) are treated as fixed. Consequently, the proposed estimator $\widehat{\mathrm{MSE}}_{mpq}(\hat{t}_I)$ does not account for the additional variability induced by the data-driven selection of these tuning parameters. This simplification may lead to an underestimation of the true mean squared error.

5.3 Estimation of the MSE: Pseudo-Population Bootstrap approach

An alternative to the analytical variance estimation approach based on the Särndal decomposition is to rely on bootstrap methods. In survey sampling, bootstrap procedures differ from those used in classical statistics due to the finite-population framework and the presence of complex sampling designs. In particular, the resampling mechanism must reproduce the original sampling design and preserve the finite-population structure.

Several bootstrap methods have been proposed for finite-population inference. One widely used approach is the pseudo-population bootstrap (PPB), originally introduced by Gross [1980] and further developed by Booth et al. [1994]. A general overview of bootstrap methods for complex survey designs can be found in Mashreghi et al. [2016]. The core idea of the pseudo-population bootstrap is to reconstruct a synthetic population from the sample, from which bootstrap samples are then repeatedly drawn in order to mimic the original sampling process.

To illustrate the construction, consider the case of simple random sampling without replacement (SRSWOR). Suppose that a population of size $N = 2000$ is sampled to obtain a sample S of size $n = 200$. A pseudo-population U^* of size N can be constructed by replicating each sampled unit approximately N/n times. When the ratio N/n is an integer, each sampled unit can simply be replicated exactly N/n times. In most practical situations, however, N/n is not an integer. To address this issue, Booth et al. [1994] proposed a randomized completion procedure. Each sampled unit is first replicated $k = \lfloor N/n \rfloor$ times, forming a fixed component of size nk. The pseudo-population is then completed by selecting an additional simple random sample of size $N - nk$ without replacement from the original sample. Other randomization mechanisms have also been proposed, for instance by Bickel and Freedman [1984] and Chao and Lo [1985].

Following the general description in Mashreghi et al. [2016], the pseudo-population bootstrap algorithm for SRSWOR can be summarized as follows.

1. Construct a pseudo-population U^* of size N by replicating sampled units according to the procedure described above.

2. Draw a bootstrap sample S^* of size n without replacement from U^* using the same sampling design as in the original survey.

3. Compute the estimator of interest from the bootstrap sample and denote it by $\hat{\theta}^*$.

4. Repeat the previous steps B times to obtain bootstrap replicates $\hat{\theta}^*_1, \ldots, \hat{\theta}^*_B$.

The bootstrap variance estimator is then obtained as

$$\widehat{\mathbb{V}}^*_B = \frac{1}{B-1} \sum_{b=1}^{B} \left(\hat{\theta}^*_b - \hat{\theta}^*_{(\cdot)} \right)^2, \tag{5.3.1}$$

where $\hat{\theta}^*_{(\cdot)} = B^{-1} \sum_{b=1}^{B} \hat{\theta}^*_b$. Similarly, a bootstrap estimator of the mean squared error can be defined as

$$\widehat{\mathrm{MSE}}^*_B = \frac{1}{B} \sum_{b=1}^{B} \left(\hat{\theta}^*_b - \theta^*_b \right)^2, \tag{5.3.2}$$

where θ^*_b denotes the parameter computed from the bth pseudo-population.

The algorithm described above corresponds to the standard pseudo-population bootstrap under the assumption that all sampled values are fully observed, that is, in the absence of nonresponse. In this case, the estimator $\hat{\theta}^*$ can be computed directly from each bootstrap sample S^* without any additional step. In the presence of item nonresponse, however, this procedure must be adapted to account for the imputation process. Following the approach proposed by Shao and Sitter [1996], the response status of each unit is kept fixed across bootstrap samples: a respondent in the original sample remains a respondent in each bootstrap sample, and a nonrespondent remains a nonrespondent. Thus, each bootstrap sample S^* is partitioned into a respondent set S^*_r and a nonrespondent set S^*_m that mirror the response pattern observed in the original sample.

For each bootstrap sample, the imputation procedure must then be re-applied. More precisely, the imputation model is re-estimated using the bootstrap respondent set S^*_r, and the missing values for units in S^*_m are re-imputed using exactly the same imputation method as in the original sample. In the case of ridge-type imputation, this implies that the regression coefficients are re-estimated within each bootstrap sample and that the penalty parameter λ is re-selected at each bootstrap iteration. The selection of λ may be carried out either by cross-validation or by using the CMSE criterion described in Chapter 4.

This re-imputation step ensures that the bootstrap replicates properly reflect the additional variability introduced by the estimation of the imputation model and by the selection of the regularization parameter. That is, unlike the analytical approach described in the previous section, the bootstrap naturally captures the additional variability induced by the data-driven selection of the tuning parameters.

5.4 Empirical Results

In this section, we consider two simulation experiments. In the first one, we evaluate the performance of the MSE estimator based on the approach of Sarndal [1992]; see Section 5.4.1 . In the second one (see Section 5.4.2), we assess the performance of an MSE estimator based on a pseudo-population bootstrap procedure. In both cases, the simulation setup remains identical to that of Section 4.7. The main difference is that the focus here is no longer on the accuracy of the imputed means, but rather on the accuracy of the proposed MSE estimators.

5.4.1 MSE estimation based on the approach of Särndal (1992)

In this section, we assess the performance of MSE estimator given by (5.2.18) based on the approach of Sarndal [1992]. Specifically, we considered:

- the OLS imputation estimator,
- the Ridge regression imputation estimator (with λ selected via the CMSE criterion),
- the debiased Ridge estimators, including the first-order debiased estimator (DR-1) and the data-driven estimator (BEST), where both the tuning parameter λ and the order of debiasing are selected using the CMSE criterion.

In all cases, the error variance σ^2 is estimated using the usual unbiased OLS estimator based on the respondents,

$$\widehat{\sigma}^2 = \frac{\sum_{i \in S_r} (y_i - \mathbf{x}_i^\top \widehat{\beta}_r)^2}{n_r - p}.$$

We performed 5,000 Monte Carlo iterations, for each Monte Carlo iteration $j = 1, \ldots, J$, in addition to the point estimate $\hat{\mu}^{(m)}$, we compute its corresponding Särndal MSE estimator, denoted as $\widehat{\text{MSE}}_{mpq}(\hat{\mu}^{(m)})$.

As a measure of bias of the MSE estimator, $\widehat{\text{MSE}}_{mpq}(\hat{\mu}^{(m)})$, we computed its Monte Carlo percent relative bias (RB):

$$\text{RB}\left(\widehat{\text{MSE}}_{mpq}(\hat{\mu}^{(m)})\right) = 100 \times \frac{\frac{1}{J}\sum_{j=1}^{J} \widehat{\text{MSE}}_{mpq}(\hat{\mu}^{(m)})^{(j)} - \text{MSE}_{MC}(\hat{\mu}^{(m)})}{\text{MSE}_{MC}(\hat{\mu}^{(m)})}, \tag{5.4.1}$$

where $\text{MSE}_{MC}(\hat{\mu}^{(m)}) = \frac{1}{J}\sum_{j=1}^{J}(\hat{\mu}^{(j)} - \mu_{true})^2$.

Tables 5.1 and 5.2 report the Monte Carlo percent relative bias of the Särndal-based MSE estimator under moderate (corr = 0.3) and strong (corr = 0.95) correlation among predictors.

When the correlation is moderate (corr = 0.3), the OLS MSE estimator behaves well overall. For example, when $p/\mathbb{E}(n_r) = 30/100$, the relative bias is only about 2.8%, and it stays below 6% even for larger values of $p/\mathbb{E}(n_r)$. In contrast, the ridge-based estimators tend to slightly

Table 5.1: Monte Carlo percent relative bias of the MSE estimator based on the approach of Särndal, for corr = 0.3.

Estimator	$p/\mathbb{E}(n_r)$				
	$\frac{10}{100}$	$\frac{20}{100}$	$\frac{30}{100}$	$\frac{40}{100}$	$\frac{50}{100}$
OLS	2.7	1.6	2.8	5.7	5.9
Ridge (CMSE)	-0.8	-6.6	-8.8	-9.0	-9.9
DR-1 (CMSE)	-3.5	-12.1	-12.1	-10.5	-11.4
BEST (CMSE)	-0.7	-6.5	-6.6	-5.7	-6.6

Table 5.2: Monte Carlo percent relative bias of the MSE estimator based on the approach of Särndal, corr = 0.95.

Estimator	$p/\mathbb{E}(n_r)$				
	$\frac{10}{100}$	$\frac{20}{100}$	$\frac{30}{100}$	$\frac{40}{100}$	$\frac{50}{100}$
OLS	4.1	4.4	5.0	4.2	8.4
Ridge (CMSE)	1.9	2.5	1.1	0.6	2.1
DR-1 (CMSE)	0.2	0.1	-0.9	-1.9	-1.0
BEST (CMSE)	1.9	2.5	1.3	0.6	1.9

underestimate the MSE. This is already visible at $p/\mathbb{E}(n_r) = 20/100$ (around -6.6%) and becomes somewhat more noticeable as $p/\mathbb{E}(n_r)$ increases (about -9.9% at 50/100), although these biases remain moderate in magnitude. The DR-1 estimator shows a similar pattern, with a negative bias across all settings (e.g., about -12.1% at 30/100). The BEST estimator is slightly more stable, although it also tends to underestimate the MSE.

When the correlation is strong (corr = 0.95), the behavior is better. The OLS estimator remains accurate; for example, the bias is around 5.0% at 30/100 and does not increase much afterward. The ridge and BEST estimators also perform well, with biases typically between 0% and 2.5%. The DR-1 estimator is now much more accurate than in the moderate-correlation case, with biases close to zero (e.g., about -0.9% at 30/100).

Overall, the observed biases are relatively small for MSE estimators and remain within an acceptable range in most settings. This is not unexpected, as estimating the MSE is inherently more challenging than estimating a mean, and deviations of a few percentage points are generally considered reasonable in this context.

5.4.2 MSE estimation based on the pseudo-population bootstrap procedure

In this section, we assess the performance of the bootstrap procedure described in Section 5.3 in terms of percent relative bias. For each Monte Carlo iteration, we applied the bootstrap procedure described in Section 5.3 with $B = 250$ bootstrap samples. Given the computational cost of the bootstrap procedure, we performed $J = 1{,}000$ Monte Carlo iterations instead of 5,000. The bootstrap was used to estimate the MSE of the following imputed estimators: OLS, Ridge with λ selected by CMSE, Adaptive Lasso, Post-Lasso, and the debiased Ridge estimator with the best pair (λ, m) selected using the CMSE criterion (denoted BEST).

The results reported in Tables 5.3 and 5.4 show that the bootstrap MSE estimator performs reasonably well when the dimension is small, but its behavior quickly deteriorates as $p/\mathbb{E}(n_r)$ increases.

When corr = 0.3, the bootstrap performs adequately for small values of $p/\mathbb{E}(n_r)$. For instance, at 10/100 and 20/100, the relative bias remains moderate for most estimators (e.g., -2.5% and 2.9% for OLS). However, as $p/\mathbb{E}(n_r)$ increases, the behavior becomes unstable. In particular, for OLS, the bootstrap strongly overestimates the MSE, with a bias of 57.9% at 40/100 and exploding to more than 16,000% at 50/100. Similar, although less extreme, patterns are observed for the other estimators. Overall, the bootstrap tends to break down in high-dimensional settings.

A similar pattern is observed when corr = 0.95. For small values of $p/\mathbb{E}(n_r)$, the bootstrap remains reasonably accurate. However, as the dimension increases, the bias becomes very

large. For example, the OLS bootstrap MSE has a bias of 68.3% at 40/100 and exceeds 120,000% at 50/100.

This severe overestimation is consistent with known results in the literature (e.g., El Karoui and Purdom, 2018), which show that classical bootstrap methods can fail in high-dimensional regimes.

Table 5.3: Monte Carlo percent relative bias of the MSE estimator based on the pseudo-population bootstrap, for corr = 0.3.

Estimator	$p/\mathbb{E}(n_r)$				
	$\frac{10}{100}$	$\frac{20}{100}$	$\frac{30}{100}$	$\frac{40}{100}$	$\frac{50}{100}$
OLS	-2.5	2.9	16.6	57.9	16346.6
Ridge (CMSE)	-4.5	-6.5	-3.6	3.2	38.2
A-LASSO	-9.3	-23.5	-25.0	-17.8	-13.5
P-LASSO	-2.1	2.5	10.3	23.9	25.3
BEST (CMSE)	-4.6	-8.1	-9.0	-5.9	34.9

Table 5.4: Monte Carlo percent relative bias of the MSE estimator based on the pseudo-population bootstrap, for corr = 0.95.

Estimator	$p/\mathbb{E}(n_r)$				
	$\frac{10}{100}$	$\frac{20}{100}$	$\frac{30}{100}$	$\frac{40}{100}$	$\frac{50}{100}$
OLS	-5.0	2.4	19.6	68.3	124243.1
Ridge (CMSE)	-8.4	-7.4	-7.9	1.8	27.7
A-LASSO	-10.1	-7.9	2.4	9.6	8.5
P-LASSO	-3.1	7.7	13.5	35.2	33.0
BEST (CMSE)	-8.6	-7.9	-8.4	1.2	37.7

Chapter 6

Final Remarks

In this thesis, we studied the use of ridge regression for imputation in a survey sampling context. We also considered debiased versions of the ridge estimator and proposed a method to select both the order m and the penalty parameter λ. Our ridge estimation approach based on minimizing the conditional mean squared error (CMSE), which is a natural criterion in this setting.

Overall, the results show that the proposed approach performs well across a range of scenarios. However, we were not able to identify situations where CMSE-based criterion clearly outperforms more standard approaches, such as selecting the penalty parameter using cross-validation. This suggests that while the CMSE-based selection is theoretically well motivated, its practical advantage over existing methods may be limited in some settings.

We also proposed two methods for estimating the mean squared error of the imputed estimators. The first is based on a linearization approach inspired by Sarndal [1992], and the second relies on a pseudo-population bootstrap. The linearization-based estimator performed well across all scenarios considered, providing reliable estimates of the MSE for both ridge and debiased ridge imputation.

The bootstrap approach also gave reasonable results when the ratio p/n is small. However, in high-dimensional settings, it tends to overestimate the MSE. This behavior is consistent with what has been observed in the i.i.d. setting (see, e.g., El Karoui and Purdom [2018], where bootstrap methods can perform poorly when the dimension is large relative to the sample size. One possible direction for future research would be to develop bootstrap procedures for survey sampling that are more robust to high-dimensional settings. This is a challenging problem, but it could lead to more reliable variance estimation methods in practice.

This thesis focused on linear regression models for imputation. For future work, several additional simulation scenarios could be explored to provide a more comprehensive assessment of ridge and debiased ridge imputation methods. One might be interested in alternative

correlation structures for the covariates, such as block-equicorrelated structures, to capture different forms of multicollinearity. One may also study on the model robustness, generating $\mathbf{X}$ using non-Gaussian covariates from heavy-tailed or skewed distributions, such as multivariate t_ν or log-normal model. Finally, since ridge and LASSO-type methods are also available for generalized linear models, extending the proposed methodology to this broader class of models would be a natural and interesting direction for future study.

Bibliography

Zhidong Bai and Jack W Silverstein. *Spectral Analysis of Large Dimensional Random Matrices*. Springer Series in Statistics. Springer, New York, 2nd edition, 2010.

P. Bardsley and R. L. Chambers. Multipurpose estimation from unbalanced samples. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 33(3):290–299, 1984.

Jean-François Beaumont and Cynthia Bocci. Another look at ridge calibration. *Metron*, 66 (1):5–20, 2008.

Jean-François Beaumont and Cynthia Bocci. Variance estimation when donor imputation is used to fill in missing values. *Canadian Journal of Statistics*, 37(3):400–416, 2009.

Alexandre Belloni and Victor Chernozhukov. Least squares after model selection in high-dimensional sparse models. *Bernoulli*, 19(2):521–547, 2013.

Alexandre Belloni, Victor Chernozhukov, and Christian Hansen. Inference on treatment effects after selection among high-dimensional controls. *The Review of Economic Studies*, 79(2):608–650, 2014.

P. J. Bickel and D. A. Freedman. Asymptotic normality and the bootstrap in stratified sampling. *The Annals of Statistics*, 12(2):470–482, 1984.

J. G. Booth, R. W. Butler, and P. Hall. Bootstrap methods for finite populations. *Journal of the American Statistical Association*, 89(428):1282–1289, 1994.

F Jay Breidt and Jean D Opsomer. Model-assisted survey estimation with modern prediction techniques. *Statistical Science*, 32(2):190–205, 2017.

C. M. Cassel, C. E. Särndal, and J. H. Wretman. Some results on generalized difference estimation and generalized regression estimation for finite populations. *Biometrika*, 63(3): 615–620, 1976.

M. T. Chao and S.-H. Lo. A bootstrap method for finite population. *Sankhyā: The Indian Journal of Statistics, Series A*, 47:399–405, 1985.

Jack Kuang Tsung Chen, Richard L. Valliant, and Michael R. Elliott. Model-assisted calibration of non-probability sample survey data using adaptive lasso. *Survey Methodology*, 44(1):117–144, 2018.

Shaochuang Chen and David Haziza. Recent developments in dealing with item nonresponse in surveys: A critical review. *International Statistical Review*, 87(S1):S192–S218, 2019.

Mehdi Dagdoug, Camelia Goga, and David Haziza. Model-assisted estimation in high-dimensional settings for survey data. *Journal of Applied Statistics*, 50(3):761–785, 2023. Published online 2022.

Noureddine El Karoui and Elizabeth Purdom. Can we trust the bootstrap in high-dimensions? the case of linear models. *Journal of Machine Learning Research*, 19(5): 1–66, 2018.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010.

Zhaoxing Gao and Ruey S. Tsay. Optimal bias-correction and valid inference in high-dimensional ridge regression: A closed-form solution. *arXiv preprint arXiv:2405.00424*, 2024. URL `https://arxiv.org/abs/2405.00424`.

Camelia Goga. Overview of ridge regression estimators in survey sampling. In Shalabh and G. N. Singh, editors, *Recent Advances in Survey Sampling and P.V. Sukhatme Felicitation Volume*. Narosa Publishing House, 2011.

S. Gross. Median estimation in sample surveys. In *Proceedings of the Section on Survey Research Methods*, pages 181–184. American Statistical Association, 1980.

Fabrice Guggemos and Yves Tillé. Penalized calibration in survey sampling: Design-based estimation assisted by mixed models. *Journal of Statistical Planning and Inference*, 140 (11):3199–3212, 2010.

Arthur E. Hoerl and Robert W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

D. G. Horvitz and D. J. Thompson. A generalization of sampling without replacement from a finite universe. *Journal of the American Statistical Association*, 47(260):663–685, 1952.

Roderick JA Little and Donald B Rubin. *Statistical analysis with missing data*, volume 793. John Wiley & Sons, 2019.

Erin R. Lundy and J. N. K. Rao. Relative performance of methods based on model-assisted survey regression estimation: A simulation study. *Survey Methodology*, 48(1):1–26, 2022.

Vladimir A Marchenko and Leonid A Pastur. Distribution of eigenvalues for some sets of random matrices. *Mathematics of the USSR-Sbornik*, 1(4):457–483, 1967.

Zeinab Mashreghi, David Haziza, and Christian Léger. A survey of bootstrap methods in finite population sampling. *Statistics Surveys*, 10:1–52, 2016. doi: 10.1214/16-SS113.

Kelly S. McConville, F. Jay Breidt, Thomas C. M. Lee, and Gretchen G. Moisen. Model-assisted survey regression estimation with the lasso. *Journal of Survey Statistics and Methodology*, 5(2):131–158, 2017.

P. M. Robinson and C. E. Särndal. Asymptotic properties of the generalized regression estimator in probability sampling. *Sankhyā: The Indian Journal of Statistics, Series B*, 45(2):240–248, 1983.

D. B. Rubin. Inference and missing data. *Biometrika*, 63:581–592, 1976.

CarL-Erik Sarndal. Metholds for estimating the precision of survey estimates when imputation has been used. *Survey Methodology*, 18(2):241–252, 1992.

J. Shao and R. R. Sitter. Bootstrap for imputed survey data. *Journal of the American Statistical Association*, 91(435):1278–1288, 1996.

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

Y. Zhang and Dimitris N. Politis. Ridge regression revisited: Debiasing, thresholding and bootstrap. *Electronic Journal of Statistics*, 16(2):3866–3902, 2022. arXiv:2009.08071.

Hui Zou. The adaptive lasso and its oracle properties. *Journal of the American Statistical Association*, 101(476):1418–1429, 2006.

Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.

Appendix A

This appendix contains the detailed mathematical proofs supporting the results presented in the main document.

A.1 Proposition 2.3.1

Proof. Let:

$$\mathbf{A} = \left(\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_p\right)^{-1},$$

Then at order m ($m \geq 1$) the m th–order debiased ridge estimator adds the term

$$\lambda^{\,m}\, \mathbf{A}^{\,m+1} \mathbf{X}^\top \mathbf{y},$$

After summing to convergence the full estimator is

$$\widehat{\boldsymbol{\beta}}_{m,deb}(\lambda) \;=\; \Big[\mathbf{A} + \lambda \mathbf{A}^2 + \lambda^2 \mathbf{A}^3 + \cdots\Big] \mathbf{X}^\top \mathbf{y} \;=\; \mathbf{A}\left[\mathbf{I}_p - \lambda \mathbf{A}\right]^{-1} \mathbf{X}^\top \mathbf{y}.$$

By Neumann series. Now:

$$\mathbf{I}_p - \lambda \mathbf{A} \;=\; \mathbf{I}_p - \lambda \left(\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_p\right)^{-1} \;=\; \left(\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_p\right)^{-1} \mathbf{X}^\top \mathbf{X}.$$

Hence

$$\begin{aligned}
\left[\mathbf{I}_p - \lambda \mathbf{A}\right]^{-1} &= (\mathbf{X}^\top \mathbf{X})^{-1} \left(\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_p\right), \\
\mathbf{A}\left[\mathbf{I}_p - \lambda \mathbf{A}\right]^{-1} &= \mathbf{A} (\mathbf{X}^\top \mathbf{X})^{-1} \left(\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_p\right) \\
&= \left(\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_p\right)^{-1} (\mathbf{X}^\top \mathbf{X})^{-1} \left(\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_p\right) \\
&= (\mathbf{X}^\top \mathbf{X})^{-1}.
\end{aligned}$$

Therefore:

$$\begin{aligned}
\widehat{\boldsymbol{\beta}}_{m,deb}(\lambda) &= \mathbf{A}\left[\mathbf{I}_p - \lambda \mathbf{A}\right]^{-1} \mathbf{X}^\top \mathbf{y} \\
&= (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}, \text{ as } m \to \infty.
\end{aligned}$$

∎

A.2 Proposition 4.3.1

Proof. We have:

$$\begin{aligned}
Bias(\hat{t}_I^{lin}) &= \mathbb{E}_{mpq}(\hat{t}_I^{lin} - t_y) \\
&= \mathbb{E}_{mpq}(\hat{t}_I^{lin} - \hat{t}_{HT}) + \mathbb{E}_{mpq}(\hat{t}_{HT} - t_y) \\
&= \mathbb{E}_{mpq}(\hat{t}_I^{lin} - \hat{t}_{HT}) + 0 \\
&= \mathbb{E}_{mpq}(\hat{t}_I^{lin} - \hat{t}_{HT}) \\
&= \mathbb{E}_m\mathbb{E}_p\mathbb{E}_q(\hat{t}_I^{lin} - \hat{t}_{HT}) \\
&= \mathbb{E}_p\mathbb{E}_q\mathbb{E}_m(\sum_{i\in S} w_i r_i y_i + \sum_{i\in S} w_i(1-r_i)y^*_{i\,linear} - \sum_{i\in S} w_i y_i) \\
&= \mathbb{E}_q\mathbb{E}_p(\sum_{i\in S} w_i r_i \mathbf{x}_i^\top \boldsymbol{\beta} + \sum_{i\in S} w_i(1-r_i)\mathbf{x}_i^\top \boldsymbol{\beta} - \sum_{i\in S} w_i \mathbf{x}_i^\top \boldsymbol{\beta}) \\
&= \mathbb{E}_q(\sum_{i\in U} r_i y_i + \sum_{i\in U}(1-r_i)\mathbf{x}_i^\top \boldsymbol{\beta} - \sum_{i\in U} \mathbf{x}_i^\top \boldsymbol{\beta}) \\
&= \sum_{i\in U} p_i \mathbf{x}_i^\top \boldsymbol{\beta} + \sum_{i\in U}(1-p_i)\mathbf{x}_i^\top \boldsymbol{\beta} - \sum_{i\in U} \mathbf{x}_i^\top \boldsymbol{\beta} \\
&= \sum_{i\in U} \mathbf{x}_i^\top \boldsymbol{\beta} - \sum_{i\in U} \mathbf{x}_i^\top \boldsymbol{\beta} \\
&= \mathbf{0}
\end{aligned}$$

∎

A.3 Proof for $\hat{\mathbb{V}}_{sam}$

Proof. We have:

$$\begin{aligned}
\hat{\mathbb{V}}_{HT} &= \sum_{i\in S}\sum_{j\in S} \frac{(\pi_{ij} - \pi_i\pi_j)}{\pi_{ij}} \frac{y_i}{\pi_i}\frac{y_j}{\pi_j} \\
&= \sum_{i\in S_r}\sum_{j\in S_r} \frac{(\pi_{ij} - \pi_i\pi_j)}{\pi_{ij}} \frac{y_i}{\pi_i}\frac{y_j}{\pi_j} + \sum_{i\in S_r}\sum_{j\in S_m} \frac{(\pi_{ij} - \pi_i\pi_j)}{\pi_{ij}} \frac{y_i}{\pi_i}\frac{y_j}{\pi_j} \\
&\quad + \sum_{i\in S_m}\sum_{j\in S_r} \frac{(\pi_{ij} - \pi_i\pi_j)}{\pi_{ij}} \frac{y_i}{\pi_i}\frac{y_j}{\pi_j} + \sum_{i\in S_m}\sum_{j\in S_m} \frac{(\pi_{ij} - \pi_i\pi_j)}{\pi_{ij}} \frac{y_i}{\pi_i}\frac{y_j}{\pi_j} \\
&= \sum_{i\in S_r} \frac{(1-\pi_i)}{\pi_i^2} y_i^2 + \sum_{i\in S_r}\sum_{\substack{j\in S_r \\ i\neq j}} \frac{(\pi_{ij} - \pi_i\pi_j)}{\pi_{ij}} \frac{y_i}{\pi_i}\frac{y_j}{\pi_j} + \sum_{i\in S_r}\sum_{j\in S_m} \frac{(\pi_{ij} - \pi_i\pi_j)}{\pi_{ij}} \frac{y_i}{\pi_i}\frac{y_j}{\pi_j} \\
&\quad + \sum_{i\in S_m}\sum_{j\in S_r} \frac{(\pi_{ij} - \pi_i\pi_j)}{\pi_{ij}} \frac{y_i}{\pi_i}\frac{y_j}{\pi_j} + \sum_{i\in S_m}\sum_{\substack{j\in S_m \\ i\neq j}} \frac{(\pi_{ij} - \pi_i\pi_j)}{\pi_{ij}} \frac{y_i}{\pi_i}\frac{y_j}{\pi_j} + \sum_{i\in S_m} \frac{(1-\pi_i)}{\pi_i^2} y_i^2,
\end{aligned}$$

$$
\begin{aligned}
\mathbb{E}_m(\hat{\mathbb{V}}_{HT}|Y_r) &= \sum_{i\in S_r}\frac{1-\pi_i}{{\pi_i}^2}{y_i}^2 + \sum_{i\in S_r}\sum_{j\in S_r i\neq j}\frac{\pi_{ij}-\pi_i\pi_j}{\pi_{ij}}\frac{y_i}{\pi_i}\frac{y_j}{\pi_j} \\
&\quad + \sum_{i\in S_r}\sum_{j\in S_m}\frac{\pi_{ij}-\pi_i\pi_j}{\pi_{ij}}\frac{y_i}{\pi_i}\frac{{\mathbf{v}_j}^\top\boldsymbol{\beta}}{\pi_j} + \sum_{i\in S_m}\sum_{j\in S_r}\frac{\pi_{ij}-\pi_i\pi_j}{\pi_{ij}}\frac{{\mathbf{v}_i}^\top\boldsymbol{\beta}}{\pi_i}\frac{y_j}{\pi_j} \\
&\quad + \sum_{i\in S_m}\sum_{j\in S_m i\neq j}\frac{\pi_{ij}-\pi_i\pi_j}{\pi_{ij}}\frac{{\mathbf{v}_i}^\top\boldsymbol{\beta}}{\pi_i}\frac{{\mathbf{v}_j}^\top\boldsymbol{\beta}}{\pi_j} + \sum_{i\in S_m}\frac{1-\pi_i}{{\pi_i}^2}(({\mathbf{v}_i}^\top\boldsymbol{\beta})^2+\sigma^2) \\
&= \Bigg(\sum_{i\in S_r}\frac{1-\pi_i}{{\pi_i}^2}{y_i}^2 + \sum_{i\in S_r}\sum_{j\in S_r i\neq j}\frac{\pi_{ij}-\pi_i\pi_j}{\pi_{ij}}\frac{y_i}{\pi_i}\frac{y_j}{\pi_j} \\
&\quad + \sum_{i\in S_r}\sum_{j\in S_m}\frac{\pi_{ij}-\pi_i\pi_j}{\pi_{ij}}\frac{y_i}{\pi_i}\frac{{\mathbf{v}_j}^\top\boldsymbol{\beta}}{\pi_j} + \sum_{i\in S_m}\sum_{j\in S_r}\frac{\pi_{ij}-\pi_i\pi_j}{\pi_{ij}}\frac{{\mathbf{v}_i}^\top\boldsymbol{\beta}}{\pi_i}\frac{y_j}{\pi_j} \\
&\quad + \sum_{i\in S_m}\sum_{j\in S_m i\neq j}\frac{\pi_{ij}-\pi_i\pi_j}{\pi_{ij}}\frac{{\mathbf{v}_i}^\top\boldsymbol{\beta}}{\pi_i}\frac{{\mathbf{v}_j}^\top\boldsymbol{\beta}}{\pi_j} + \sum_{i\in S_m}\frac{1-\pi_i}{{\pi_i}^2}({\mathbf{v}_i}^\top\boldsymbol{\beta})^2\Bigg) + \sum_{i\in S_m}\frac{1-\pi_i}{{\pi_i}^2}\sigma^2, \\
&= \hat{\mathbb{V}}_{naive} + \mathbb{V}_{diff}.
\end{aligned}
$$

∎

A.4 Proof for $\hat{\mathbb{V}}_{nr}$

Proof. We have:

$$
\begin{aligned}
\hat{t}_I &= \sum_{i\in S} w_i r_i y_i + \sum_{i\in S} w_i \mathbf{v}_i^\top\hat{\boldsymbol{\beta}}_r - \sum_{i\in S} r_i w_i \mathbf{v}_i^\top\hat{\boldsymbol{\beta}}_r \\
&= \sum_{i\in S} w_i r_i y_i + \left(\sum_{i\in S} w_i\mathbf{v}_i^\top - \sum_{i\in S} r_i w_i \mathbf{v}_i^\top\right)\hat{\boldsymbol{\beta}}_r \\
&= \sum_{i\in S} w_i r_i y_i + \left(\sum_{i\in S} w_i\mathbf{v}_i^\top - \sum_{i\in S} r_i w_i \mathbf{v}_i^\top\right)\left(\sum_{i\in S_r} w_i\mathbf{v}_i\mathbf{v}_i^\top\right)^{-1}\left(\sum_{i\in S_r} w_i\mathbf{v}_i y_i\right) \\
&= \sum_{i\in S_r} w_i\left(1+\left(\sum_{i\in S} w_i\mathbf{v}_i - \sum_{i\in S_r} w_i\mathbf{v}_i\right)^\top\left(\sum_{i\in S_r} w_i\mathbf{v}_i\mathbf{v}_i^\top\right)^{-1}\mathbf{v}_i\right) y_i \\
&= \sum_{i\in S_r} w_i g_i y_i, \\
\mathbb{V}_m\left(\hat{t}_I - \hat{t}_{\mathrm{HT}}\right) &= \mathbb{V}_m\left(\sum_{i\in S_r} w_i g_i y_i - \sum_{i\in S} w_i y_i\right) \\
&= \mathbb{V}_m\left(\sum_{i\in S} w_i(r_i g_i - 1) y_i\right)
\end{aligned}
$$

$$= \sigma^2 \left(\sum_{i \in S} w_i^2 (r_i g_i - 1)^2 \right),$$

$$\mathbb{V}_{nr} = \sigma^2 \mathbb{E}_p \mathbb{E}_q \left(\sum_{i \in S} w_i^2 (r_i g_i - 1)^2 \right).$$

∎

A.5 Proof for $\hat{\mathbb{V}}_{mix}$

Proof. We have:

$$\begin{aligned}
\hat{t}_{HT} - t_y &= \sum_{i \in U} (I_i w_i - 1) y_i, \\
\hat{t}_I - \hat{t}_{HT} &= \sum_{i \in U} I_i w_i (r_i g_i - 1) y_i, \\
\mathrm{Cov}_m \left\{ \left(\hat{t}_{\mathrm{HT}} - t_y \right), \left(\hat{t}_I - \hat{t}_{\mathrm{HT}} \right) \right\} &= \mathrm{Cov}_m \left\{ \sum_{i \in U} (I_i w_i - 1) y_i, \sum_{i \in U} I_i w_i (r_i g_i - 1) y_i \right\} \\
&= \sum_{i \in U} (I_i w_i - 1) I_i w_i (r_i g_i - 1) \mathrm{Cov}_m (y_i, y_i) \\
&= \sum_{i \in S} w_i^2 (1 - \pi_i) (r_i g_i - 1) \mathrm{Cov}_m (y_i, y_i) \\
&= \left(\sigma^2 \sum_{i \in S} w_i^2 (1 - \pi_i) (r_i g_i - 1) \right), \\
\mathbb{V}_{mix} &= \sigma^2 \mathbb{E}_p \mathbb{E}_q \left(\sum_{i \in S} w_i^2 (1 - \pi_i) (r_i g_i - 1) \right).
\end{aligned}$$

∎

A.6 Source Code

A.6.1 Debiased Ridge Regression

Listing 1: Regression Simulation Script

```r
# Load necessary library
library(MASS)        # for mvrnorm()
library(glmnet)      # for ridge and adaptive lasso (cv.glmnet)
library(doParallel) # for parallel computing
library(doRNG)
library(foreach)
library(data.table)
library(hdi)

# 1. Parameters
set.seed(123)
n         <- 100      # sample size
p         <- 20       # number of predictors
R         <- 2000     # Monte Carlo reps
K_max     <- 50       # compute bias corrected ridge up to order k
lambda    <- 0.1 * n  # ridge penalty
tol       <- 1e-3     # convergence tolerance
max_iter <- 100      # hard cap on iterations

# 2. Build X once via multivariate normal with correlation 0.9
rho <- 0.95
Sigma <- matrix(rho, nrow = p, ncol = p)
diag(Sigma) <- 1  # Correlation matrix with 1's on the diagonal
sigma_eps <- 1

set.seed(123)
X <- mvrnorm(n = n, mu = rep(0, p), Sigma = Sigma)
#Standardize predictors (mean 0, SD 1)
X <- scale(X, center = TRUE, scale = FALSE)

# 3. True beta
half <- p %/% 2
beta <- c(
  runif(half, 0, 1),
  runif(half, 1, 2)
)

sv <- svd(X)                          # X = V D U'
d  <- sv$d[1:p]                       # singular values (p <= n)
U1 <- sv$v[, 1:p, drop = FALSE]       # right singular vectors

# 3.1 Calculate delta from the fixed beta
delta <- as.vector(t(U1) %*% beta)

# 3.2 Choose lambda to GUARANTEE the condition is met
# We need lambda > sigma_eps^2 / min(delta^2)
# Set lambda slightly above the boundary to avoid precision issues.
```

```r
lambda <- (sigma_eps^2 / min(delta^2)) * 1.001
cat(sprintf("Condition guaranteed by setting lambda = %.4f\n", lambda))

# 3.3 Check the condition (will now pass by construction)
condition_LHS <- delta^2
condition_RHS <- sigma_eps^2 / lambda

if (all(condition_LHS > condition_RHS)) {
  cat("Success: The condition δ_i^2 > σ^2/λ is satisfied for all i.\n")
} else {
  # This part of the code should not be reached
  cat("Failed: The condition is NOT satisfied.\n")
}

# CV.lambda
y <- as.vector(X %*% beta + rnorm(n))
y <- y - mean(y)
cv_ridge <- cv.glmnet(x = X,
                      y = y, alpha = 0, standardize = FALSE,intercept =
                          FALSE)
lambda_opt <- cv_ridge$lambda.min  # Optimal lambda (k in the formula)
#lambda <- n * lambda_opt / sd(y)

# 4. Preliminary computations
XtX  <- crossprod(X)
Ip   <- diag(p)
invM <- solve(XtX + lambda * Ip)    # = (X'X + λI)^(-1)

# 4.1 Determinant of (X'X)^{-1}
det_invXtX <- det(invM)
det_XtX<-det(XtX)
cat("Determinant of (X'X)^{-1} =", det_invXtX, "\n")
cat("Determinant of (X'X)=", det_XtX, "\n")

# 5. Storage
beta_hat_all <- array(NA, c(p, K_max+1, R))
beta_conv    <- matrix(NA, p, R)
k_iter       <- integer(R)

for(r in seq_len(R)) {

  # 5.1 simulate y
  y <- as.vector(X %*% beta + rnorm(n, sd = sigma_eps))
  y <- y - mean(y)

  # 5.2 ridge at k=0
  ridge_beta <- invM %*% crossprod(X, y)
  beta_hat_all[,1,r] <- ridge_beta

  # 5.3 bias corrected ridge up to K_max
  invM_pow <- invM
  sum_term <- lambda * (invM_pow %*% ridge_beta)
  for(k in 1:K_max) {
    if(k > 1) {
```

```
      invM_pow <- invM_pow %*% invM
      sum_term <- sum_term + (lambda^k) * (invM_pow %*% ridge_beta)
    }
    beta_hat_all[, k+1, r] <- ridge_beta + sum_term
  }

  # 5.4 "until convergence" debiased ridge
  invM_pow_c  <- invM
  sum_term_c  <- lambda * (invM_pow_c %*% ridge_beta)
  corrected_c <- ridge_beta + sum_term_c
  k_c         <- 1

  repeat {
    invM_pow_c <- invM_pow_c %*% invM
    new_term   <- (lambda^(k_c+1)) * (invM_pow_c %*% ridge_beta)
    max_abs <- suppressWarnings(max(abs(new_term), na.rm = TRUE))
    if (!is.finite(max_abs) || max_abs < tol || k_c >= max_iter) {
      # optional: message to know why you stopped
      #message("Break at k = ", k_c, "  (max_abs = ", max_abs, ")")
      break
    }
    sum_term_c  <- sum_term_c + new_term
    corrected_c <- ridge_beta + sum_term_c
    k_c         <- k_c + 1
  }

  k_iter[r]     <- k_c
  beta_conv[,r] <- as.vector(corrected_c)

}

# 6. Monte Carlo MSE for bias corrected at each k

mse_k <- sapply(0:K_max, function(k) {
  errs <- sapply(seq_len(R), function(r) {
    sum((beta_hat_all[, k+1, r] - beta)^2)
  })
  mean(errs)
})


# 7. MSE & mean k for converged estimator
sq_errs_conv <- colSums((beta_conv - beta)^2)
mse_conv     <- mean(sq_errs_conv)
mean_k       <- mean(k_iter)

cat(sprintf(
  "Converged algorithm: avg k = %.2f (over %d replicates)\n  MSE = %.4f\n",
  mean_k, R, mse_conv
))

# 8. Plot
plot(0:K_max, mse_k, type="b", pch=19, lwd=2,
```

```
     xlab="Number of iterations k",
     ylab="Monte Carlo MSE",
     main="MSE of Bias Corrected Ridge vs. k")
points(mean_k, mse_conv, col="red", pch=19, cex=1.5)
text(mean_k, mse_conv,
     labels=paste0("(",round(mean_k,1),", ",round(mse_conv,3),")"),
     pos=4, col="red")

# 9. Monte Carlo Averages of Coefficients for Each k
beta_means <- sapply(0:K_max, function(k) {
  rowMeans(beta_hat_all[, k+1, ])
})

colnames(beta_means) <- paste0("k=", 0:K_max)
rownames(beta_means) <- paste0("beta_", 1:p)
beta_means_df <- as.data.frame(beta_means)
cat("\nMonte Carlo Averages of Regression Coefficients (first 10):\n")
print(head(beta_means_df, 10))  # First 10 coefficients

# 10. Table of Monte Carlo MSEs by k
mse_table <- data.frame(
  k   = 0:K_max,
  MSE = round(mse_k, 4)
)

cat("\nMonte Carlo MSE for each value of k:\n")
print(mse_table)

OLSbeta <- rep(1, R)
coef_mat <- replicate(R, {
  y <- as.vector(X %*% beta + rnorm(n))
  y <- y - mean(y)
  coef(lm(y ~ X - 1))
})
OLSMSE <- c()
for (r in 1:R) {
  OLSMSE[r] <- sum((coef_mat[,r] - beta)^2)
}
print(mean(OLSMSE))

write(mse_k, file = "mse_k_output.txt")
```

A.6.2 Data Imputation and Sarndal MSE

Listing 2: Main Simulation and Imputation Script

```
suppressPackageStartupMessages({
  if (!requireNamespace("pbapply", quietly = TRUE)) install.packages("
     pbapply")
  if (!requireNamespace("glmnet", quietly = TRUE))  install.packages("
     glmnet")
```

```
  if (!requireNamespace("MASS", quietly = TRUE))    install.packages("MASS")
  if (!requireNamespace("Matrix", quietly = TRUE))  install.packages("Matrix")
  library(pbapply)
  library(glmnet)
  library(MASS)
  library(Matrix)
  library(parallel)
})
RNGkind(kind = "L'Ecuyer-CMRG", normal.kind = "Inversion", sample.kind = "Rejection")
pboptions(type = "txt")
set.seed(1)

# --------------------------------
# Simulation Parameters (EDIT ME)
# --------------------------------
N <- 2000
n <- 200
p <- 10
rho <- 0.3
R_squared <- 0.6
M <- 5000

# Ridge tuning options
nfolds_cv     <- 10
choose_lambda <- "min"     # "min" or "1se"

# Intercept alignment flag used internally
mean_align_intercept <- TRUE

# Lambda grid for CMSE optimization
lambda_grid_cmse <- exp(seq(log(1e-4), log(1e+2), length.out = 60))

# Adaptive Lasso hyperparams
alasso_gamma <- 1.0
alasso_eps   <- 1e-6

# Optional output directory ("" = skip saving)
target_dir <- ""

# --------------------------------
# Data generation helpers
# --------------------------------
generate_X_population <- function(N, p, rho, mu = 0) {
  Sigma <- matrix(rho, nrow = p, ncol = p); diag(Sigma) <- 1
  X <- mvrnorm(n = N, mu = rep(mu, p), Sigma = Sigma)
  colnames(X) <- paste0("X", 1:p)
  X
}

generate_beta_true <- function(p) {
```

```r
  # Calculate the number of coefficients for each part (20/20/60  or 30/30/40 or 50/50/0)
  p1_count <- round(p * 0.5)
  p2_count <- round(p * 0.5)
  p3_count <- p - p1_count - p2_count # The rest are zero

  # Generate the coefficient values for each part
  part1_betas <- runif(p1_count, 0, 1)  # small non-zero
  part2_betas <- runif(p2_count, 1, 2)  # large non-zero
  part3_betas <- rep(0, p3_count)       # sparse (zero)

  # Combine all parts into one vector
  beta_combined <- c(part1_betas, part2_betas, part3_betas)

  # Shuffle to randomize the positions of zero and non-zero coefficients
  beta_shuffled <- sample(beta_combined)

  return(beta_shuffled)
}

generate_Y_fixed <- function(X, beta_true, R_squared, intercept = 10) {
  lin <- intercept + as.numeric(X %*% beta_true)
  vl  <- var(lin); if (vl <= .Machine$double.eps) vl <- .Machine$double.eps
  ve  <- vl * (1 - R_squared) / max(R_squared, 1e-8)
  lin + rnorm(nrow(X), 0, sqrt(ve))
}

generate_missing <- function(X, target_response_rate = 0.5) {
  n <- nrow(X)
  lin <- 0.2 * rowSums(X)
  trg <- (target_response_rate - 0.1) / 0.9
  trg <- min(max(trg, 0.01), 0.99)
  a <- as.numeric(quantile(lin, probs = 1 - trg))
  pr <- 0.1 + 0.9 * (1 / (1 + exp(-(lin - a))))
  rbinom(n, 1, pr)   # 1 = respondent
}

# --------------------------------
# Generate ONCE: X_pop and beta
# --------------------------------
beta_true <- generate_beta_true(p)
X_pop     <- generate_X_population(N, p, rho)
X_pop_mm  <- as.matrix(X_pop)

cat("Predictors X and beta generated once; Y regenerated each iteration.\n")

# --------- helpers: closed-form ridge on centered data (raw scale) ----------
ridge_closed_form <- function(X, y, lambda) {
  xbar <- colMeans(X); ybar <- mean(y)
  Xc <- sweep(X, 2, xbar, "-")
  yc <- y - ybar
```

```r
  XtX <- crossprod(Xc)
  XtY <- crossprod(Xc, yc)
  A   <- XtX + diag(lambda, ncol(XtX))
  R   <- chol(A)
  beta <- backsolve(R, forwardsolve(t(R), XtY))
  intercept <- as.numeric(ybar - sum(xbar * beta))
  list(beta = as.numeric(beta), intercept = intercept,
       XtX = XtX, Xc = Xc, yc = yc, xbar = xbar, ybar = ybar)
}

# Intercept aligned to respondents (used internally)
align_intercept <- function(beta_vec, X_resp, y_resp) {
  ybar <- mean(y_resp)
  xbar <- colMeans(X_resp)
  as.numeric(ybar - sum(xbar * beta_vec))
}

# Debiased ridge slopes via repeated solves: beta_DR(k) = beta_R + sum_{j
    =1..k} λ^j A^{-j} beta_R
debias_beta_k <- function(Rchol, beta_r, lambda, k) {
  solve_A <- function(v) backsolve(Rchol, forwardsolve(t(Rchol), v))
  v  <- solve_A(beta_r)   # A^{-1} beta_R
  s  <- lambda * v
  if (k >= 2) {
    for (j in 2:k) {
      v <- solve_A(v)     # A^{-j} beta_R
      s <- s + (lambda^j) * v
    }
  }
  beta_r + s
}

# Utility: apply A^{-m} to a vector by repeated solves (m >= 1)
Ainv_power_vec <- function(Rchol, vec, m) {
  out <- vec
  for (i in 1:m) out <- backsolve(Rchol, forwardsolve(t(Rchol), out))
  out
}

# Utility: L_k(vec) = sum_{j=0..k} λ^j A^{-(j+1)} vec
Lk_apply_vec <- function(Rchol, vec, lambda, k) {
  out <- Ainv_power_vec(Rchol, vec, 1)  # j=0 term: A^{-1} vec
  if (k >= 1) {
    v <- out
    for (j in 1:k) {
      v <- backsolve(Rchol, forwardsolve(t(Rchol), v))  # A^{-(j+1)} vec
      out <- out + (lambda^j) * v
    }
  }
  out
}

# --------- Original CMSE plug-in for DR(k), k>=0 (k=0=ridge) ----------
cmse_hat_drk <- function(lambda, k, XtX, XtY, Xc, yc, Dx, nR, nNR) {
```

```r
  n <- nR + nNR
  if (nNR <= 0) return(0)
  cfrac <- nNR / n
  p <- ncol(Xc)

  A <- XtX + diag(lambda, p)
  Rchol <- chol(A)

  # Core ridge slope
  beta_r <- backsolve(Rchol, forwardsolve(t(Rchol), XtY))  # A^{-1} XtY

  # Squared bias (plug-in)
  Ainvk1_beta <- Ainv_power_vec(Rchol, beta_r, k + 1)
  bdot <- as.numeric(crossprod(Dx, Ainvk1_beta))
  term_bias2 <- (lambda^(2*(k+1))) * (bdot^2)

  # Variance direction
  v_Dx <- Lk_apply_vec(Rchol, Dx, lambda, k)  # L_k Dx
  var_dir_core <- as.numeric(crossprod(v_Dx, XtX %*% v_Dx))

  # σ^2 via ridge df at this λ
  Ainv <- chol2inv(Rchol)
  df_ridge <- sum(XtX * Ainv)
  denom <- max(1e-8, (nR - df_ridge))
  sigma2_hat <- sum((yc - Xc %*% beta_r)^2) / denom

  # Covariance term calculation
  # Formula: (2 * sigma^2 / nR) * (Delta_x^T D_m sum(x_i))
  sum_Xc <- colSums(Xc)
  term_cov <- (2 * sigma2_hat / nR) * sum(v_Dx * sum_Xc)

  # Final return including term_cov
  (cfrac^2) * ( term_bias2 + sigma2_hat * var_dir_core + sigma2_hat * (1/nR
      + 1/nNR) + term_cov )
}

# --------- General CMSE function with external plug-ins ----------
cmse_general_plug <- function(lambda, k, XtX, Dx, nR, nNR, beta_plug,
    sigma2_plug, sum_Xc) {

  n <- nR + nNR
  if (nNR <= 0) return(0)
  cfrac <- nNR / n
  p <- ncol(XtX)

  A <- XtX + diag(lambda, p)
  Rchol <- chol(A)

  # 1. Squared bias term: Uses external beta_plug
  # Bias^2 = (lambda^(m+1) * Delta_x^T * A^{-(m+1)} * beta_plug)^2
  Ainvk1_beta <- Ainv_power_vec(Rchol, beta_plug, k + 1)
  bdot <- as.numeric(crossprod(Dx, Ainvk1_beta))
  term_bias2 <- (lambda^(2*(k+1))) * (bdot^2)

```

```
  # 2. Variance term: Uses external sigma2_plug
  # Var = sigma^2 * Delta_x^T * L_m * XtX * L_m^T * Delta_x
  v_Dx <- Lk_apply_vec(Rchol, Dx, lambda, k)
  var_dir_core <- as.numeric(crossprod(v_Dx, XtX %*% v_Dx))

  # 3. Covariance term
  # Cov = (2 * sigma^2 / nR) * (Delta_x^T D_m sum(x_i))
  term_cov <- (2 * sigma2_plug / nR) * sum(v_Dx * sum_Xc)

  # Combine
  (cfrac^2) * ( term_bias2 + sigma2_plug * var_dir_core + sigma2_plug * (1/
     nR + 1/nNR) + term_cov )
}

# --------------------------------
# Särndal MSE Decomposition Function
# --------------------------------
calc_sarndal_mse <- function(X_s,      # (n x p) Sample design matrix (
   intercept included if needed)
                             y_resp,   # (n_r x 1) Respondent y values
                             R_ind,    # (n x 1) Response indicator (1=
                                Resp, 0=NonResp)
                             beta_hat, # (p x 1) The coefficients used for
                                 imputation (OLS, Ridge, or DR)
                             A_mat_inv,# (p x p) Inverse of A matrix for g
                                -weights (Inverse of XtX+lamI)
                             N_pop,    # Population size (for scaling to
                                Mean)
                             sigma2_input = NULL # Optional: plug-in sigma
                                2
) {

  # 1. Setup Dimensions and Weights for SRSWOR
  n <- nrow(X_s)
  n_resp <- length(y_resp)
  w_i <- N_pop / n        # Design weight (Uniform for SRS)
  pi_i <- n / N_pop       # Inclusion probability
  fpc <- 1 - pi_i         # Finite population correction factor (1 - n/N)

  # Separate X
  X_resp <- X_s[R_ind == 1, , drop = FALSE]

  # 2. Imputed Values (y_tilde) for the whole sample S
  # y_tilde = y_i if R=1, else x_i'beta if R=0
  y_hat_s <- as.numeric(X_s %*% beta_hat)
  y_s_imp <- numeric(n)
  y_s_imp[R_ind == 1] <- y_resp
  y_s_imp[R_ind == 0] <- y_hat_s[R_ind == 0]

  # 3. Estimate Sigma^2 (if not provided)
  # Typically uses respondent residuals.
  if (is.null(sigma2_input)) {
    resid_r <- y_resp - as.numeric(X_resp %*% beta_hat)
    # Note: Using weighted sum of squares / sum of weights
```

```
    # For SRS, w_i cancels out, effectively mean(resid^2)
    sigma2_hat <- sum(w_i * resid_r^2) / sum(rep(w_i, n_resp))
  } else {
    sigma2_hat <- sigma2_input
  }

  # ------------------------------------------------------------
  # Component 1: V_sam (Sampling Variance)
  # Eq: V_sam = V_naive + V_diff
  # ------------------------------------------------------------

  # A. V_naive: Standard HT Variance on Imputed Data y_tilde
  # For SRSWOR: V_HT(y_bar) = (1-f)/n * S^2_y_tilde
  y_bar_imp <- mean(y_s_imp)
  S2_y_imp  <- var(y_s_imp) # Sample variance of imputed data
  # Variance of TOTAL = N^2 * (1-f)/n * S^2
  V_naive_total <- (N_pop^2) * (1 - n/N_pop) / n * S2_y_imp

  # B. V_diff: Correction for treating imputed as observed
  # For SRS: w_i = N/n, pi = n/N. Term is constant for non-respondents.
  # Sum over S of (1-r_i) is n_nr.
  # V_diff = sigma^2 * (N/n)^2 * (1 - n/N) * n_nr
  n_nr <- n - n_resp
  term_diff <- (w_i^2) * (1 - pi_i) * n_nr
  V_diff_total <- sigma2_hat * term_diff

  V_sam_total <- V_naive_total + V_diff_total

  # ------------------------------------------------------------
  # Calculate g-weights (g_i) for V_nr, V_mix, V_extra
  # g_i = 1 + (Sum_S w x - Sum_Sr w x)' A^-1 x_i
  # ------------------------------------------------------------
  # Calculate totals
  T_X_S  <- colSums(w_i * X_s)      # Estimated Pop Total from Sample S
  T_X_Sr <- colSums(w_i * X_resp)   # Estimated Pop Total from Respondents
  diff_X <- T_X_S - T_X_Sr          # Basically Total of Non-respondents (
     imputed part)

  # Compute term: diff_X' A^-1
  # A_mat_inv is passed in. For OLS it's (X'X)^-1. For Ridge it's (X'X +
     lam I)^-1
  vec_proj <- as.numeric(crossprod(diff_X, A_mat_inv)) # 1 x p

  # g_i for ALL i in S
  # inner product of vec_proj and x_i
  # g_i = 1 + X_s %*% t(vec_proj)
  g_vec <- 1 + as.numeric(X_s %*% vec_proj)

  # Helper term used in V_nr, V_mix, V_extra: (r_i * g_i - 1)
  # r_i is 1 or 0.
  # If r=1: (g_i - 1). If r=0: (-1).
  rg_minus_1 <- (R_ind * g_vec) - 1

  # ------------------------------------------------------------
```

```r
  # Component 2: V_nr (Non-response Variance)
  # sigma^2 * sum_{s} w_i^2 * (r_i g_i - 1)^2
  # ------------------------------------------------------------
  term_nr <- sum( (w_i^2) * (rg_minus_1^2) )
  V_nr_total <- sigma2_hat * term_nr

  # ------------------------------------------------------------
  # Component 3: V_mix (Interaction Term)
  # sigma^2 * sum_{s} w_i^2 * (1-pi) * (r_i g_i - 1)
  # ------------------------------------------------------------
  # Note: sum over S
  term_mix <- sum( (w_i^2) * (1 - pi_i) * rg_minus_1 )
  V_mix_total <- sigma2_hat * term_mix

  # ------------------------------------------------------------
  # Component 4: V_extra (Squared Bias Correction)
  # [ sum_{s} w_i (r_i g_i - 1) x_i' beta_hat ]^2
  # ------------------------------------------------------------
  # Note: This checks if the weighted sum of predictions balances out.
  # For OLS, this should be effectively zero (numerically).
  # For Ridge/DR, this captures the bias.

  # w_i * (rg - 1)
  w_rg_1 <- w_i * rg_minus_1

  # We need sum( w_rg_1 * y_hat_s )
  # y_hat_s is X beta
  sum_bias <- sum(w_rg_1 * y_hat_s)
  V_extra_total <- sum_bias^2

  # ------------------------------------------------------------
  # Total MSE (Target: Total)
  # V_sam + V_nr + 2*V_mix + V_extra
  # ------------------------------------------------------------
  MSE_total_est <- V_sam_total + V_nr_total + 2*V_mix_total + V_extra_
     total

  # ------------------------------------------------------------
  # Convert to Mean Scale (divide by N^2)
  # ------------------------------------------------------------
  scale_factor <- 1 / (N_pop^2)

  list(
    sarndal_V_sam   = V_sam_total * scale_factor,
    sarndal_V_nr    = V_nr_total * scale_factor,
    sarndal_V_mix   = V_mix_total * scale_factor,
    sarndal_V_extra = V_extra_total * scale_factor,
    sarndal_MSE     = MSE_total_est * scale_factor,
    sigma2_used     = sigma2_hat
  )
}

calc_sarndal_mse_deb_ridge <- function(X_s,          # (n x p) Sample
   design matrix (intercept included)
```

```
                                        y_resp,       # (n_r x 1)
                                           Respondent y values
                                        R_ind,        # (n x 1) Response
                                           indicator
                                        lambda_raw,   # The RAW lambda (
                                           unscaled)
                                        m_order,      # Debiasing order (e.
                                           g., 1)
                                        beta_hat, # (p x 1) The
                                           coefficients used for imputation
                                        beta_ols,     # (p x 1) The OLS
                                           estimator (beta_hat in Vextra)
                                        A_mat_inv,    # (p x p) Generic
                                           Inverse for g-weights
                                        N_pop,        # Population size
                                        sigma2_input = NULL # Optional:
                                           plug-in sigma^2
) {

  # 1. Setup Dimensions and Weights for SRSWOR
  n <- nrow(X_s)
  n_resp <- length(y_resp)
  w_i <- N_pop / n         # Design weight (Uniform for SRS)
  pi_i <- n / N_pop        # Inclusion probability
  fpc <- 1 - pi_i          # Finite population correction factor

  # Separate X
  X_resp <- X_s[R_ind == 1, , drop = FALSE]

  # 2. Imputed Values (y_tilde) -> Used for V_naive S^2 calculation
  y_hat_s <- as.numeric(X_s %*% beta_hat)
  y_s_imp <- numeric(n)
  y_s_imp[R_ind == 1] <- y_resp
  y_s_imp[R_ind == 0] <- y_hat_s[R_ind == 0]

  # 3. Estimate Sigma^2 (if not provided)
  if (is.null(sigma2_input)) {
    resid_r <- y_resp - as.numeric(X_resp %*% beta_hat)
    sigma2_hat <- sum(w_i * resid_r^2) / sum(rep(w_i, n_resp))
  } else {
    sigma2_hat <- sigma2_input
  }

  # ------------------------------------------------------------
  # Component 1: V_sam (Sampling Variance)
  # ------------------------------------------------------------

  # A. V_naive
  y_bar_imp <- mean(y_s_imp)
  S2_y_imp  <- var(y_s_imp)
  V_naive_total <- (N_pop^2) * (1 - n/N_pop) / n * S2_y_imp

  # B. V_diff
  n_nr <- n - n_resp
```

```
406   term_diff <- (w_i^2) * (1 - pi_i) * n_nr
407   V_diff_total <- sigma2_hat * term_diff
408
409   V_sam_total <- V_naive_total + V_diff_total
410
411   # ------------------------------------------------------------
412   # Calculate g-weights (g_i) for V_nr, V_mix
413   # ------------------------------------------------------------
414   T_X_S  <- colSums(w_i * X_s)
415   T_X_Sr <- colSums(w_i * X_resp)
416   diff_X <- T_X_S - T_X_Sr
417
418   vec_proj <- as.numeric(crossprod(diff_X, A_mat_inv))
419   g_vec <- 1 + as.numeric(X_s %*% vec_proj)
420   rg_minus_1 <- (R_ind * g_vec) - 1
421
422   # ------------------------------------------------------------
423   # Component 2: V_nr (Non-response Variance)
424   # ------------------------------------------------------------
425   term_nr <- sum( (w_i^2) * (rg_minus_1^2) )
426   V_nr_total <- sigma2_hat * term_nr
427
428   # ------------------------------------------------------------
429   # Component 3: V_mix (Interaction Term)
430   # ------------------------------------------------------------
431   term_mix <- sum( (w_i^2) * (1 - pi_i) * rg_minus_1 )
432   V_mix_total <- sigma2_hat * term_mix
433
434   # ------------------------------------------------------------
435   # Component 4: V_extra using explicit Bias formula
436   # Bias = -lambda^(m+1) * (Sum_Sm w x)' * (Ar + lam I)^-(m+1) * beta_ols
437   # ------------------------------------------------------------
438
439   # 1. The Vector Part: Sum_{i in Sm} w_i * x_i
440   # Sum over Non-Respondents (Sm)
441   X_miss <- X_s[R_ind == 0, , drop = FALSE]
442   if (nrow(X_miss) > 0) {
443     vec_sum_miss <- colSums(w_i * X_miss) # Dimension (p x 1)
444   } else {
445     vec_sum_miss <- rep(0, ncol(X_s))
446   }
447
448   # 2. The Matrix Part: (Ar + lambda*I)^-(m+1)
449   # Ar = Sum_{Sr} w_j * x_j * x_j^T
450   Ar_weighted <- crossprod(X_resp) * w_i
451   lam_weighted <- lambda_raw * w_i
452
453   # Construct (Ar + lam_weighted * I)
454   # Assuming first column is intercept and is NOT penalized
455   penalty_mat <- diag(rep(lam_weighted, ncol(X_s)))
456   penalty_mat[1,1] <- 0
457
458   Mat_base <- Ar_weighted + penalty_mat
459   Mat_inv  <- chol2inv(chol(Mat_base)) # This is (Ar + lam I)^-1
```

```r

  # Compute Power: (Mat_inv)^(m+1)
  Mat_power <- Mat_inv
  iter_count <- m_order + 1
  if (iter_count > 1) {
    for(k in 2:iter_count) {
      Mat_power <- Mat_power %*% Mat_inv
    }
  }

  # 3. The Scalar Part: - lambda^(m+1)
  term_scalar <- -1 * (lambda_raw)^(m_order + 1)

  # 4. Calculate Bias
  # Bias = Scalar * (Vector)^T * Matrix * Beta_OLS
  dot_prod <- as.numeric( crossprod(vec_sum_miss, Mat_power %*% beta_ols) )
  bias_est_total <- term_scalar * dot_prod

  # 5. V_extra is Squared Bias
  V_extra_total <- bias_est_total^2

  # ------------------------------------------------------------
  # Total MSE (Target: Total)
  # ------------------------------------------------------------
  MSE_total_est <- V_sam_total + V_nr_total + 2*V_mix_total + V_extra_total

  # ------------------------------------------------------------
  # Convert to Mean Scale (divide by N^2)
  # ------------------------------------------------------------
  scale_factor <- 1 / (N_pop^2)

  list(
    sarndal_V_sam   = V_sam_total * scale_factor,
    sarndal_V_nr    = V_nr_total * scale_factor,
    sarndal_V_mix   = V_mix_total * scale_factor,
    sarndal_V_extra = V_extra_total * scale_factor, # Now using Eq 20 Bias^2
    sarndal_MSE     = MSE_total_est * scale_factor,
    sigma2_used     = sigma2_hat,
    bias_est        = bias_est_total * sqrt(scale_factor) # Output explicit bias for debugging
  )
}

# ------------------------------
# One Monte Carlo iteration
# ------------------------------
one_mc <- function(iter) {
  # New finite population outcomes and true mean
  Y_pop  <- generate_Y_fixed(X_pop_mm, beta_true, R_squared, intercept = 10)
  mu_true <- mean(Y_pop)
```

```r

  # SRSWOR sample
  s_idx <- sample.int(N, size = n, replace = FALSE)
  Xs <- X_pop_mm[s_idx, , drop = FALSE]
  ys <- Y_pop[s_idx]

  # Nonresponse on the sample
  R <- generate_missing(Xs, target_response_rate = 0.5)
  if (sum(R) < max(3L, p + 2L)) {
    R <- generate_missing(Xs, target_response_rate = 0.5)
  }

  # Split
  X_resp <- Xs[R == 1, , drop = FALSE]
  y_resp <- ys[R == 1]
  n_resp <- length(y_resp)
  X_non  <- Xs[R == 0, , drop = FALSE]
  n_non  <- n - n_resp

  # Means and centered pieces for respondents
  xbar_R <- colMeans(X_resp)
  ybar_R <- mean(y_resp)
  xbar_NR <- if (n_non > 0) colMeans(X_non) else rep(0, p)
  Dx <- as.numeric(xbar_NR - xbar_R)
  Xc <- sweep(X_resp, 2, xbar_R, "-")
  yc <- as.numeric(y_resp - ybar_R)
  XtX <- crossprod(Xc)
  XtY <- crossprod(Xc, yc)

  # ---------- (A) OLS imputation ----------
  fit_ols <- lm(y ~ ., data = data.frame(y = y_resp, X_resp))
  yhat_ols <- as.numeric(predict(fit_ols, newdata = data.frame(Xs)))
  y_tilde_ols <- ys; y_tilde_ols[R == 0] <- yhat_ols[R == 0]
  mu_imp_ols <- mean(y_tilde_ols)

  # ---------- (B) Ridge (CV-λ) ----------
  nf <- min(nfolds_cv, max(2L, n_resp))
  cvfit <- cv.glmnet(
    x = X_resp, y = y_resp,
    alpha = 0,
    nfolds = nf,
    standardize = FALSE,
    intercept = TRUE,
    family = "gaussian",
    type.measure = "mse",
    parallel = FALSE
  )
  lam_cv <- if (choose_lambda == "1se") cvfit$lambda.1se else cvfit$lambda
     .min
  lam_cv <- lam_cv*nrow(X_resp)/sd(y_resp)
  # Closed-form ridge (raw scale) at lam_cv
  cf   <- ridge_closed_form(X_resp, y_resp, lambda = lam_cv)
  beta_r <- cf$beta
  b0_r_align <- align_intercept(beta_r, X_resp, y_resp)
```

```r
  yhat_ridge <- as.numeric(b0_r_align + Xs %*% beta_r)
  y_tilde_ridge <- ys; y_tilde_ridge[R == 0] <- yhat_ridge[R == 0]
  mu_imp_ridge <- mean(y_tilde_ridge)

  # ---------- Adaptive Lasso (CV-λ) ----------
  w_alasso <- 1 / (abs(beta_r) + alasso_eps)^alasso_gamma
  cvfit_alasso <- cv.glmnet(
    x = X_resp, y = y_resp,
    alpha = 1,
    nfolds = nf,
    standardize = FALSE,
    intercept = TRUE,
    family = "gaussian",
    type.measure = "mse",
    penalty.factor = w_alasso,
    parallel = FALSE
  )
  lam_alasso <- cvfit_alasso$lambda.min*nrow(X_resp)/sd(y_resp)
  coef_alasso <- as.matrix(coef(cvfit_alasso, s = "lambda.min"))
  beta_alasso <- as.numeric(coef_alasso[-1, , drop = TRUE])
  b0_alasso <- align_intercept(beta_alasso, X_resp, y_resp)
  yhat_alasso <- as.numeric(b0_alasso + Xs %*% beta_alasso)
  y_tilde_alasso <- ys; y_tilde_alasso[R == 0] <- yhat_alasso[R == 0]
  mu_imp_alasso <- mean(y_tilde_alasso)

  # ---------- Post-OLS Lasso (support from Adaptive Lasso) ----------
  sel_idx <- which(abs(beta_alasso) > 0)
  if (length(sel_idx) == 0L) {
    beta_post <- rep(0, p)
    b0_post   <- mean(y_resp)
  } else {
    X_resp_sel <- X_resp[, sel_idx, drop = FALSE]
    fit_post <- lm(y ~ ., data = data.frame(y = y_resp, X_resp_sel))
    b_post <- coef(fit_post)
    beta_post <- rep(0, p)
    beta_post[sel_idx] <- as.numeric(b_post[-1])
    b0_post <- align_intercept(beta_post, X_resp, y_resp)
  }
  yhat_post <- as.numeric(b0_post + Xs %*% beta_post)
  y_tilde_post <- ys; y_tilde_post[R == 0] <- yhat_post[R == 0]
  mu_imp_postlasso <- mean(y_tilde_post)

  # ---------- (C) Debiased Ridge k=1..5 at CV-λ ----------
  A_cv   <- XtX + diag(lam_cv, p)
  Rchol_cv  <- chol(A_cv)
  beta_r_cv <- backsolve(Rchol_cv, forwardsolve(t(Rchol_cv), XtY))
  beta_dr1 <- debias_beta_k(Rchol_cv, beta_r_cv, lam_cv, k = 1)
  beta_dr2 <- debias_beta_k(Rchol_cv, beta_r_cv, lam_cv, k = 2)
  beta_dr3 <- debias_beta_k(Rchol_cv, beta_r_cv, lam_cv, k = 3)
  beta_dr4 <- debias_beta_k(Rchol_cv, beta_r_cv, lam_cv, k = 4)
  beta_dr5 <- debias_beta_k(Rchol_cv, beta_r_cv, lam_cv, k = 5)

  imp_mean_with_beta <- function(beta_vec) {
    b0_used <- align_intercept(beta_vec, X_resp, y_resp)
```

```
    yhat <- as.numeric(b0_used + Xs %*% beta_vec)
    ytil <- ys; ytil[R == 0] <- yhat[R == 0]
    mean(ytil)
  }
  mu_imp_dr1 <- imp_mean_with_beta(beta_dr1)
  mu_imp_dr2 <- imp_mean_with_beta(beta_dr2)
  mu_imp_dr3 <- imp_mean_with_beta(beta_dr3)
  mu_imp_dr4 <- imp_mean_with_beta(beta_dr4)
  mu_imp_dr5 <- imp_mean_with_beta(beta_dr5)

  # ---------- (D) Ridge (CMSE-λ) ----------
  if (n_non > 0) {
    cmse_vals <- vapply(lambda_grid_cmse, function(lam) {
      cmse_hat_drk(lam, 0L, XtX, XtY, Xc, yc, Dx, n_resp, n_non)
    }, numeric(1))
    lam_cmse <- lambda_grid_cmse[which.min(cmse_vals)]

    cf_cmse <- ridge_closed_form(X_resp, y_resp, lambda = lam_cmse)
    beta_cmse <- cf_cmse$beta
    b0_cmse_used <- if (mean_align_intercept) align_intercept(beta_cmse, X
       _resp, y_resp) else cf_cmse$intercept
    yhat_cmse <- as.numeric(b0_cmse_used + Xs %*% beta_cmse)
    y_tilde_cmse <- ys; y_tilde_cmse[R == 0] <- yhat_cmse[R == 0]
    mu_imp_cmse <- mean(y_tilde_cmse)
  } else {
    lam_cmse <- NA_real_
    mu_imp_cmse <- mean(ys)
  }

  beta_ols_plug <- coef(fit_ols)[-1]
  beta_ols_plug[is.na(beta_ols_plug)] <- 0
  sigma2_ols_plug <- (summary(fit_ols)$sigma)^2

  # 2. True Plug-in (Oracle)
  lin_pop <- 10 + as.numeric(X_pop_mm %*% beta_true) # intercept doesn't
     affect variance
  vl_pop  <- var(lin_pop)
  if (vl_pop <= .Machine$double.eps) vl_pop <- .Machine$double.eps
  sigma2_true_plug <- vl_pop * (1 - R_squared) / max(R_squared, 1e-8)

  # 3. sum_Xc for  Covariance
  sum_Xc <- colSums(Xc)


  # ---------- (E) DRk (CMSE-λ) ----------
  cmse_pick_and_impute_drk <- function(k) {
    if (n_non <= 0) return(list(mu = mean(ys), lam = NA_real_))
    vals <- vapply(lambda_grid_cmse, function(lam) {
      cmse_hat_drk(lam, k, XtX, XtY, Xc, yc, Dx, n_resp, n_non)
    }, numeric(1))
    lam_best <- lambda_grid_cmse[which.min(vals)]
    A_k <- XtX + diag(lam_best, p)
    Rchol_k <- chol(A_k)
    beta_r_k <- backsolve(Rchol_k, forwardsolve(t(Rchol_k), XtY))
```

```
    beta_drk <- if (k == 0L) beta_r_k else debias_beta_k(Rchol_k, beta_r_k
        , lam_best, k = k)
    b0_k <- if (mean_align_intercept) align_intercept(beta_drk, X_resp, y_
        resp) else {
      cf_tmp <- ridge_closed_form(X_resp, y_resp, lambda = lam_best); cf_
          tmp$intercept
    }
    yhat <- as.numeric(b0_k + Xs %*% beta_drk)
    ytil <- ys; ytil[R == 0] <- yhat[R == 0]
    list(mu = mean(ytil), lam = lam_best)
  }

  dr1 <- cmse_pick_and_impute_drk(1L)
  dr2 <- cmse_pick_and_impute_drk(2L)
  dr3 <- cmse_pick_and_impute_drk(3L)
  dr4 <- cmse_pick_and_impute_drk(4L)
  dr5 <- cmse_pick_and_impute_drk(5L)


  solve_cmse_plugin <- function(k, beta_in, sigma2_in) {
    if (n_non <= 0) return(list(mu = mean(ys), lam = NA_real_))

    vals <- vapply(lambda_grid_cmse, function(lam) {
      cmse_general_plug(lam, k, XtX, Dx, n_resp, n_non, beta_in, sigma2_in
          , sum_Xc)
    }, numeric(1))

    lam_best <- lambda_grid_cmse[which.min(vals)]

    A_k <- XtX + diag(lam_best, p)
    Rchol_k <- chol(A_k)
    beta_r_k <- backsolve(Rchol_k, forwardsolve(t(Rchol_k), XtY))

    beta_final <- if (k == 0L) beta_r_k else debias_beta_k(Rchol_k, beta_r
        _k, lam_best, k = k)

    b0_final <- if (mean_align_intercept) align_intercept(beta_final, X_
        resp, y_resp) else {
      cf_tmp <- ridge_closed_form(X_resp, y_resp, lambda = lam_best); cf_
          tmp$intercept
    }

    yhat <- as.numeric(b0_final + Xs %*% beta_final)
    ytil <- ys; ytil[R == 0] <- yhat[R == 0]
    return(mean(ytil))
  }


  # === (G) OLS-based CMSE Selection (k=1..5) ===
  mu_imp_dr1_cmse_ols <- solve_cmse_plugin(1L, beta_ols_plug, sigma2_ols_
      plug)
  mu_imp_dr2_cmse_ols <- solve_cmse_plugin(2L, beta_ols_plug, sigma2_ols_
      plug)
```

```r
mu_imp_dr3_cmse_ols <- solve_cmse_plugin(3L, beta_ols_plug, sigma2_ols_plug)
mu_imp_dr4_cmse_ols <- solve_cmse_plugin(4L, beta_ols_plug, sigma2_ols_plug)
mu_imp_dr5_cmse_ols <- solve_cmse_plugin(5L, beta_ols_plug, sigma2_ols_plug)

# === (H) True-based CMSE Selection (Oracle, k=1..5) ===
mu_imp_dr1_cmse_true <- solve_cmse_plugin(1L, beta_true, sigma2_true_plug)
mu_imp_dr2_cmse_true <- solve_cmse_plugin(2L, beta_true, sigma2_true_plug)
mu_imp_dr3_cmse_true <- solve_cmse_plugin(3L, beta_true, sigma2_true_plug)
mu_imp_dr4_cmse_true <- solve_cmse_plugin(4L, beta_true, sigma2_true_plug)
mu_imp_dr5_cmse_true <- solve_cmse_plugin(5L, beta_true, sigma2_true_plug)


# ---------- (F) BEST (k, λ) by CMSE ----------
if (n_non > 0) {
  ks <- 0:5
  best <- list(val = Inf, k = NA_integer_, lam = NA_real_)
  for (kk in ks) {
    vals <- vapply(lambda_grid_cmse, function(lam) {
      cmse_hat_drk(lam, kk, XtX, XtY, Xc, yc, Dx, n_resp, n_non)
    }, numeric(1))
    idx_min <- which.min(vals)
    if (vals[idx_min] < best$val) {
      best$val <- vals[idx_min]
      best$k   <- kk
      best$lam <- lambda_grid_cmse[idx_min]
    }
  }
  k_hat   <- best$k
  lam_hat <- best$lam

  A_best <- XtX + diag(lam_hat, p)
  Rchol_best <- chol(A_best)
  beta_r_best <- backsolve(Rchol_best, forwardsolve(t(Rchol_best), XtY))
  beta_best <- if (k_hat == 0L) beta_r_best else debias_beta_k(Rchol_best, beta_r_best, lam_hat, k = k_hat)
  b0_best <- if (mean_align_intercept) align_intercept(beta_best, X_resp, y_resp) else {
    cf_tmp <- ridge_closed_form(X_resp, y_resp, lambda = lam_hat); cf_tmp$intercept
  }
  yhat_best <- as.numeric(b0_best + Xs %*% beta_best)
  y_tilde_best <- ys; y_tilde_best[R == 0] <- yhat_best[R == 0]
  mu_imp_best <- mean(y_tilde_best)
} else {
  k_hat <- 0L; lam_hat <- NA_real_; mu_imp_best <- mean(ys)
}
```

```r


# ---------- (F) BEST (k, λ) by CMSE OLS/TRUE ----------
find_best_cmse_plugin <- function(beta_in, sigma2_in) {
  if (n_non <= 0) return(mean(ys))
  ks <- 0:5
  best_val <- Inf
  best_res <- mean(ys) # default

  for (kk in ks) {
    vals <- vapply(lambda_grid_cmse, function(lam) {
      cmse_general_plug(lam, kk, XtX, Dx, n_resp, n_non, beta_in, sigma2_in, sum_Xc)
    }, numeric(1))

    min_v <- min(vals)
    if (min_v < best_val) {
      best_val <- min_v
      lam_best <- lambda_grid_cmse[which.min(vals)]
      A_k <- XtX + diag(lam_best, p)
      Rchol_k <- chol(A_k)
      beta_r_k <- backsolve(Rchol_k, forwardsolve(t(Rchol_k), XtY))
      beta_f <- if (kk == 0L) beta_r_k else debias_beta_k(Rchol_k, beta_r_k, lam_best, k = kk)
      b0_f <- if (mean_align_intercept) align_intercept(beta_f, X_resp, y_resp) else {
        cf <- ridge_closed_form(X_resp, y_resp, lambda = lam_best); cf$intercept
      }
      yhat <- as.numeric(b0_f + Xs %*% beta_f)
      ytil <- ys; ytil[R==0] <- yhat[R==0]
      best_res <- mean(ytil)
    }
  }
  return(best_res)
}

mu_imp_best_ols  <- find_best_cmse_plugin(beta_ols_plug, sigma2_ols_plug)
mu_imp_best_true <- find_best_cmse_plugin(beta_true, sigma2_true_plug)


# Pre-calculate common terms for Sarndal
# We need Xs (sample X with intercept column) to match beta dimension
Xs_int <- cbind(1, Xs) # Add intercept column
X_resp_int <- Xs_int[R == 1, , drop=FALSE]

# For sigma^2 plug-in: Use OLS sigma^2 (most robust/consistent for variance components)
# Or use sigma2_true_plug (Oracle) if you want to test theoretical formula
sig2_plug <- sigma2_ols_plug
w_design <- N / n

```

```
# ------------------------------------------------------
# 1. Sarndal MSE for OLS
# ------------------------------------------------------
# A matrix for OLS is just X'X (weighted if needed, but here weights
    uniform)
XtX_ols <- crossprod(X_resp_int)
A_ols_inv_raw <- chol2inv(chol(XtX_ols))
A_ols_inv_weighted <- A_ols_inv_raw / w_design

beta_ols_full <- coef(fit_ols) # Extract from lm object (Intercept +
    slopes)
beta_ols_full[is.na(beta_ols_full)] <- 0

res_sarndal_ols <- calc_sarndal_mse(
  X_s = Xs_int,
  y_resp = y_resp,
  R_ind = R,
  beta_hat = beta_ols_full,
  A_mat_inv = A_ols_inv_weighted,
  N_pop = N,
  sigma2_input = sig2_plug
)

# ------------------------------------------------------
# 2. Sarndal MSE for Ridge (CMSE-optimal lambda)
# ------------------------------------------------------
# Use the BEST CMSE lambda found: lam_hat
if (!is.na(lam_cmse)) {

  XtX_int <- crossprod(X_resp_int)
  penalty_mat_cmse <- diag(rep(lam_cmse, ncol(Xs_int)))
  penalty_mat_cmse[1,1] <- 0

  A_ridge_cmse <- XtX_int + penalty_mat_cmse
  A_inv_cmse_raw <- chol2inv(chol(A_ridge_cmse))
  A_inv_cmse_weighted <- A_inv_cmse_raw / w_design

  beta_ridge_cmse_full <- A_inv_cmse_raw %*% crossprod(X_resp_int, y_
      resp)

  res_sarndal_ridge <- calc_sarndal_mse_deb_ridge(
    X_s = Xs_int,
    y_resp = y_resp,
    R_ind = R,
    lambda_raw = lam_cmse,
    m_order = 0,
    beta_hat = beta_ridge_cmse_full,
    beta_ols = beta_ols_full,
    A_mat_inv = A_inv_cmse_weighted,
    N_pop = N,
    sigma2_input = sig2_plug
  )
} else {
  res_sarndal_ridge <- list(sarndal_MSE=NA)
```

```
}

# ------------------------------------------------------
# 3. Sarndal MSE for Debiased Ridge (Best k)
# ------------------------------------------------------
if (!is.na(lam_hat)) {

  XtX_int <- crossprod(X_resp_int)
  penalty_mat_best <- diag(rep(lam_hat, ncol(Xs_int)))
  penalty_mat_best[1,1] <- 0

  A_ridge_best <- XtX_int + penalty_mat_best
  A_inv_best_raw <- chol2inv(chol(A_ridge_best))
  A_eff_inv_raw <- A_inv_best_raw


  if (k_hat > 0) {
    current_term <- A_inv_best_raw
    for (j in 1:k_hat) {
      current_term <- A_inv_best_raw %*% (penalty_mat_best %*% current_
          term)
      A_eff_inv_raw <- A_eff_inv_raw + current_term
    }
  }

  A_eff_inv_weighted <- A_eff_inv_raw / w_design

  b0_dr <- align_intercept(beta_best, X_resp, y_resp)
  beta_dr_full <- c(b0_dr, beta_best)

  res_sarndal_dr <- calc_sarndal_mse_deb_ridge(
    X_s = Xs_int,
    y_resp = y_resp,
    R_ind = R,
    lambda_raw = lam_hat,
    m_order = k_hat,
    beta_hat = beta_dr_full,
    beta_ols = beta_ols_full,
    A_mat_inv = A_eff_inv_weighted,
    N_pop = N,
    sigma2_input = sig2_plug
  )
} else {
  res_sarndal_dr <- list(sarndal_MSE=NA)
}

# ------------------------------------------------------
# 4.Sarndal MSE for First-Order Debiased Ridge (k=1)
# ------------------------------------------------------
lam_k1 <- dr1$lam

if (!is.na(lam_k1)) {
  XtX_int <- crossprod(X_resp_int)
  penalty_mat_k1 <- diag(rep(lam_k1, ncol(Xs_int)))
```

```
    penalty_mat_k1[1,1] <- 0

    A_ridge_k1_raw <- XtX_int + penalty_mat_k1
    A_ridge_k1_inv_raw <- chol2inv(chol(A_ridge_k1_raw))

    A_ridge_k1_inv_weighted <- A_ridge_k1_inv_raw / w_design

    beta_ridge_k1 <- A_ridge_k1_inv_raw %*% crossprod(X_resp_int, y_resp)

    bias_correction <- A_ridge_k1_inv_raw %*% (penalty_mat_k1 %*% beta_
        ridge_k1)
    beta_dr1_full <- beta_ridge_k1 + bias_correction

    res_sarndal_dr1 <- calc_sarndal_mse(
      X_s = Xs_int,
      y_resp = y_resp,
      R_ind = R,
      beta_hat = beta_dr1_full,
      A_mat_inv = A_ridge_k1_inv_weighted, # Use Ridge geometry for g-
          weights
      N_pop = N,
      sigma2_input = sig2_plug
    )
  } else {
    res_sarndal_dr1 <- list(sarndal_MSE = NA)
  }

  # ------------------------------------------------------
  # 5. Sarndal MSE for DR(k=1) using Explicit Bias Formula
  # ------------------------------------------------------
  if (!is.na(lam_k1)) {

    res_sarndal_dr1_explicit <- calc_sarndal_mse_deb_ridge(
      X_s = Xs_int,
      y_resp = y_resp,
      R_ind = R,
      lambda = lam_k1,
      m_order = 1,           # First order
      beta_hat = beta_dr1_full,
      beta_ols = beta_ols_full, # Use OLS beta as plug-in for Vextra
      A_mat_inv = A_ridge_k1_inv_weighted, # Use Ridge geometry for g-
          weights
      N_pop = N,
      sigma2_input = sig2_plug
    )
  } else {
    res_sarndal_dr1_explicit <- list(sarndal_MSE = NA)
  }

  # Reference: full-data sample mean (ignoring missingness)
  mu_full <- mean(ys)

  c(
    mu_imp_ols        = mu_imp_ols,
```

```
    mu_imp_ridge      = mu_imp_ridge,
    mu_imp_alasso     = mu_imp_alasso,
    mu_imp_postlasso  = mu_imp_postlasso,
    mu_imp_dr1        = mu_imp_dr1,
    mu_imp_dr2        = mu_imp_dr2,
    mu_imp_dr3        = mu_imp_dr3,
    mu_imp_dr4        = mu_imp_dr4,
    mu_imp_dr5        = mu_imp_dr5,
    mu_imp_cmse       = mu_imp_cmse,
    mu_imp_dr1_cmse   = dr1$mu,
    mu_imp_dr2_cmse   = dr2$mu,
    mu_imp_dr3_cmse   = dr3$mu,
    mu_imp_dr4_cmse   = dr4$mu,
    mu_imp_dr5_cmse   = dr5$mu,
    mu_imp_best       = mu_imp_best,

    # --- NEW: OLS-based CMSE results ---
    mu_imp_dr1_cmse_ols   = mu_imp_dr1_cmse_ols,
    mu_imp_dr2_cmse_ols   = mu_imp_dr2_cmse_ols,
    mu_imp_dr3_cmse_ols   = mu_imp_dr3_cmse_ols,
    mu_imp_dr4_cmse_ols   = mu_imp_dr4_cmse_ols,
    mu_imp_dr5_cmse_ols   = mu_imp_dr5_cmse_ols,
    mu_imp_best_ols       = mu_imp_best_ols,

    # --- NEW: True-based (Oracle) CMSE results ---
    mu_imp_dr1_cmse_true  = mu_imp_dr1_cmse_true,
    mu_imp_dr2_cmse_true  = mu_imp_dr2_cmse_true,
    mu_imp_dr3_cmse_true  = mu_imp_dr3_cmse_true,
    mu_imp_dr4_cmse_true  = mu_imp_dr4_cmse_true,
    mu_imp_dr5_cmse_true  = mu_imp_dr5_cmse_true,
    mu_imp_best_true      = mu_imp_best_true,

    # --- NEW: Sarndal MSE_results ---
    S_MSE_OLS   = res_sarndal_ols$sarndal_MSE,
    S_MSE_Ridge = res_sarndal_ridge$sarndal_MSE,
    S_MSE_DR    = res_sarndal_dr$sarndal_MSE,
    S_MSE_DR1   = res_sarndal_dr1$sarndal_MSE,
    S_MSE_DR1_Exp = res_sarndal_dr1_explicit$sarndal_MSE,

    k_hat             = k_hat,
    lambda_hat        = lam_hat,
    mu_full           = mu_full,
    mu_true           = mu_true,
    n_resp            = n_resp,
    lambda_cv         = lam_cv,
    lambda_cmse       = lam_cmse,
    lambda_alasso     = lam_alasso
  )
}

# -------------------------------
# Parallel run with libs on workers
# -------------------------------
n_cores <- max(1L, min(detectCores(logical = FALSE) - 1L, 30L))
```

```
cat("Using", n_cores, "cores.\n")
if (nzchar(target_dir)) cat("CSV will be saved to:", target_dir, "\n") else cat("No CSV output.\n")

cl <- makeCluster(n_cores)
on.exit(try(stopCluster(cl), silent = TRUE), add = TRUE)
parallel::clusterSetRNGStream(cl, 1)

# Load packages on each worker
clusterEvalQ(cl, {
  suppressPackageStartupMessages({
    library(glmnet); library(MASS); library(Matrix)
  })
  NULL
})

clusterExport(
  cl,
  varlist = c("N","n","p","rho","R_squared","M",
              "X_pop_mm","beta_true",
              "nfolds_cv","choose_lambda",
              "mean_align_intercept","lambda_grid_cmse",
              "alasso_gamma","alasso_eps",
              "generate_Y_fixed","generate_missing",
              "ridge_closed_form","align_intercept",
              "debias_beta_k","Ainv_power_vec","Lk_apply_vec",
              "cmse_general_plug",
              "cmse_hat_drk","one_mc","calc_sarndal_mse","calc_sarndal_mse_deb_ridge"),
  envir = environment()
)

res_list <- pbapply::pblapply(1:M, function(i) one_mc(i), cl = cl)
res_mat  <- do.call(rbind, res_list)

# -------------------------------
# Summaries (w.r.t. per-iter mu_true)
# -------------------------------
extract <- function(name) res_mat[, name, drop = TRUE]

mu_ols         <- extract("mu_imp_ols")
mu_ridge       <- extract("mu_imp_ridge")
mu_alasso      <- extract("mu_imp_alasso")
mu_postlasso   <- extract("mu_imp_postlasso")

mu_dr1         <- extract("mu_imp_dr1")
mu_dr2         <- extract("mu_imp_dr2")
mu_dr3         <- extract("mu_imp_dr3")
mu_dr4         <- extract("mu_imp_dr4")
mu_dr5         <- extract("mu_imp_dr5")

mu_cmse        <- extract("mu_imp_cmse")
mu_dr1_cmse    <- extract("mu_imp_dr1_cmse")
mu_dr2_cmse    <- extract("mu_imp_dr2_cmse")
```

```
mu_dr3_cmse    <- extract("mu_imp_dr3_cmse")
mu_dr4_cmse    <- extract("mu_imp_dr4_cmse")
mu_dr5_cmse    <- extract("mu_imp_dr5_cmse")

# --- [NEW] OLS-based CMSE ---
mu_dr1_cmse_ols <- extract("mu_imp_dr1_cmse_ols")
mu_dr2_cmse_ols <- extract("mu_imp_dr2_cmse_ols")
mu_dr3_cmse_ols <- extract("mu_imp_dr3_cmse_ols")
mu_dr4_cmse_ols <- extract("mu_imp_dr4_cmse_ols")
mu_dr5_cmse_ols <- extract("mu_imp_dr5_cmse_ols")
mu_best_ols      <- extract("mu_imp_best_ols")

# --- [NEW] True-based (Oracle) CMSE ---
mu_dr1_cmse_true <- extract("mu_imp_dr1_cmse_true")
mu_dr2_cmse_true <- extract("mu_imp_dr2_cmse_true")
mu_dr3_cmse_true <- extract("mu_imp_dr3_cmse_true")
mu_dr4_cmse_true <- extract("mu_imp_dr4_cmse_true")
mu_dr5_cmse_true <- extract("mu_imp_dr5_cmse_true")
mu_best_true      <- extract("mu_imp_best_true")

mu_best       <- extract("mu_imp_best")
k_hat_vec     <- extract("k_hat")
lambda_hat_vec<- extract("lambda_hat")

mu_full       <- extract("mu_full")
mu_true       <- extract("mu_true")

lambda_cv_vec     <- extract("lambda_cv")
lambda_cmse_vec   <- extract("lambda_cmse")
lambda_alasso_vec <- extract("lambda_alasso")

summarize <- function(est, truth) {
  mc_bias <- mean(est - truth)
  mc_prb  <- 100 * mc_bias / mean(truth)
  mc_mse  <- mean((est - truth)^2)
  mc_mean <- mean(est)
  c(MC_Mean = mc_mean, MC_Bias = mc_bias, MC_PctRelBias = mc_prb, MC_MSE =
      mc_mse)
}

add_row <- function(lst, label, est, truth) {
  s <- summarize(est, truth)
  lst[[length(lst) + 1L]] <- c(Estimator = label, s)
  lst
}

rows <- list()
rows <- add_row(rows, "OLS-Imputed Mean",                              mu_ols
   ,       mu_true)

rows <- add_row(rows, "Ridge (CV-λ)",                                  mu_
   ridge,     mu_true)
rows <- add_row(rows, "Adaptive Lasso (CV-λ)",                         mu_
   alasso,    mu_true)
```

```
rows <- add_row(rows, "Post-OLS Lasso (from Adaptive Lasso)",       mu_
    postlasso, mu_true)

rows <- add_row(rows, "Debiased Ridge k=1 (CV-λ)",                  mu_dr1
    ,       mu_true)
rows <- add_row(rows, "Debiased Ridge k=2 (CV-λ)",                  mu_dr2
    ,       mu_true)
rows <- add_row(rows, "Debiased Ridge k=3 (CV-λ)",                  mu_dr3
    ,       mu_true)
rows <- add_row(rows, "Debiased Ridge k=4 (CV-λ)",                  mu_dr4
    ,       mu_true)
rows <- add_row(rows, "Debiased Ridge k=5 (CV-λ)",                  mu_dr5
    ,       mu_true)

rows <- add_row(rows, "Ridge (CMSE-λ)",                             mu_
    cmse,       mu_true)
rows <- add_row(rows, "DR k=1 (CMSE-λ)",                            mu_dr1
    _cmse,  mu_true)
rows <- add_row(rows, "DR k=2 (CMSE-λ)",                            mu_dr2
    _cmse,  mu_true)
rows <- add_row(rows, "DR k=3 (CMSE-λ)",                            mu_dr3
    _cmse,  mu_true)
rows <- add_row(rows, "DR k=4 (CMSE-λ)",                            mu_dr4
    _cmse,  mu_true)
rows <- add_row(rows, "DR k=5 (CMSE-λ)",                            mu_dr5
    _cmse,  mu_true)
rows <- add_row(rows, "BEST (k, λ) by CMSE (k in 0..5)",            mu_
    best,       mu_true)

# --- [NEW] OLS-Based CMSE ---
rows <- add_row(rows, "DR k=1 (CMSE-OLS)",                  mu_dr1_cmse_
    ols, mu_true)
rows <- add_row(rows, "DR k=2 (CMSE-OLS)",                  mu_dr2_cmse_
    ols, mu_true)
rows <- add_row(rows, "DR k=3 (CMSE-OLS)",                  mu_dr3_cmse_
    ols, mu_true)
rows <- add_row(rows, "DR k=4 (CMSE-OLS)",                mu_dr4_cmse_ols,
     mu_true)
rows <- add_row(rows, "DR k=5 (CMSE-OLS)",                mu_dr5_cmse_ols,
     mu_true)
rows <- add_row(rows, "BEST (k, λ) by OLS CMSE (k in 0..5)",            mu
    _best_ols,      mu_true)

# --- [NEW] True-Based (Oracle) CMSE ---
rows <- add_row(rows, "DR k=1 (CMSE-Oracle)",               mu_dr1_cmse_
    true, mu_true)
rows <- add_row(rows, "DR k=2 (CMSE-Oracle)",               mu_dr2_cmse_
    true, mu_true)
rows <- add_row(rows, "DR k=3 (CMSE-Oracle)",               mu_dr3_cmse_
    true, mu_true)
rows <- add_row(rows, "DR k=4 (CMSE-Oracle)",             mu_dr4_cmse_true
    , mu_true)
rows <- add_row(rows, "DR k=5 (CMSE-Oracle)",             mu_dr5_cmse_true
    , mu_true)
```

```
rows <- add_row(rows, "BEST (k, λ) by TRUE CMSE (k in 0..5)",
    mu_best_true,      mu_true)


rows <- add_row(rows, "Full-Data Sample Mean",                        mu_
    full,      mu_true)

S <- do.call(rbind, rows)
S <- as.data.frame(S, stringsAsFactors = FALSE)
num_cols <- c("MC_Mean","MC_Bias","MC_PctRelBias","MC_MSE")
S[, num_cols] <- lapply(S[, num_cols, drop = FALSE], function(z) as.
    numeric(z))

out <- data.frame(
  Estimator      = S$Estimator,
  MC_Mean        = S$MC_Mean,
  Mean_TrueMean  = mean(mu_true),
  MC_Bias        = S$MC_Bias,
  MC_PctRelBias  = S$MC_PctRelBias,
  MC_MSE         = S$MC_MSE,
  check.names = FALSE
)

cat("\nMonte Carlo Results (Original CMSE only):\n")
print(format(out, digits = 6), row.names = FALSE)

# 1. Monte Carlo MSE
mc_mse_ols   <- out[out$Estimator == "OLS-Imputed Mean", "MC_MSE"]
mc_mse_ridge <- out[out$Estimator == "Ridge (CMSE-λ)", "MC_MSE"]
mc_mse_dr1   <- out[out$Estimator == "DR k=1 (CMSE-λ)", "MC_MSE"]
mc_mse_best  <- out[out$Estimator == "BEST (k, λ) by CMSE (k in 0..5)", "
    MC_MSE"]
mc_mse_dr1_exp <- mc_mse_dr1

# 2.Average Särndal MSE
mean_sarndal_ols   <- mean(res_mat[, "S_MSE_OLS"], na.rm = TRUE)
mean_sarndal_ridge <- mean(res_mat[, "S_MSE_Ridge"], na.rm = TRUE)
mean_sarndal_dr1   <- mean(res_mat[, "S_MSE_DR1"], na.rm = TRUE)
mean_sarndal_best  <- mean(res_mat[, "S_MSE_DR"], na.rm = TRUE)
mean_sarndal_dr1_exp <- mean(res_mat[, "S_MSE_DR1_Exp"], na.rm = TRUE)

# 3.DataFrame
mse_comparison <- data.frame(
  Estimator = c("OLS", "Ridge (CMSE)", "Debiased Ridge (k=1)", "Best
      Debiased Ridge", "DR (k=1, Explicit Bias)"),
  MonteCarlo_MSE = c(mc_mse_ols, mc_mse_ridge, mc_mse_dr1, mc_mse_best, mc
      _mse_dr1_exp),
  Avg_Sarndal_MSE = c(mean_sarndal_ols, mean_sarndal_ridge, mean_sarndal_
      dr1, mean_sarndal_best, mean_sarndal_dr1_exp)
)

# 4.Relative Bias of MSE Estimator
mse_comparison$Rel_Bias_MSE <- (mse_comparison$Avg_Sarndal_MSE - mse_
    comparison$MonteCarlo_MSE) / mse_comparison$MonteCarlo_MSE
```

```
1189
1190 mse_comparison_print <- mse_comparison
1191 mse_comparison_print$Rel_Bias_MSE <- sprintf("%.2f%%", mse_comparison$Rel_
        Bias_MSE * 100)
1192 cat("\nEvaluation of Särndal MSE Estimator Accuracy:\n")
1193 print(mse_comparison_print)
1194
1195 stopCluster(cl)
```

A.6.3 Bootstrapping MSE

Listing 3: Bootstrapping MSE Script

```
1  suppressPackageStartupMessages({
2    if (!requireNamespace("pbapply", quietly = TRUE)) install.packages("
        pbapply")
3    if (!requireNamespace("glmnet", quietly = TRUE))  install.packages("
        glmnet")
4    if (!requireNamespace("MASS", quietly = TRUE))    install.packages("MASS
        ")
5    if (!requireNamespace("Matrix", quietly = TRUE))  install.packages("
        Matrix")
6    library(pbapply)
7    library(glmnet)
8    library(MASS)
9    library(Matrix)
10   library(parallel)
11 })
12 RNGkind(kind = "L'Ecuyer-CMRG", normal.kind = "Inversion", sample.kind = "
       Rejection")
13 pboptions(type = "txt")
14 set.seed(1)
15
16 # --------------------------------
17 # Simulation Parameters
18 # --------------------------------
19 N <- 2000
20 n <- 200
21 p <- 10
22 rho <- 0.3
23 R_squared <- 0.6
24 M <- 1000
25
26 # Bootstrap settings
27 B_boot <- 250  # Number of bootstrap replicates per Monte Carlo iteration
28
29 # Ridge tuning options
30 nfolds_cv     <- 10
31 choose_lambda <- "min"
32 mean_align_intercept <- TRUE
33 lambda_grid_cmse <- exp(seq(log(1e-4), log(1e+2), length.out = 60))
34
```

```r
# Adaptive Lasso hyperparams
alasso_gamma <- 1.0
alasso_eps   <- 1e-6

target_dir <- ""

# ------------------------------
# Data Generation Helpers
# ------------------------------
generate_X_population <- function(N, p, rho, mu = 0) {
  Sigma <- matrix(rho, nrow = p, ncol = p); diag(Sigma) <- 1
  X <- mvrnorm(n = N, mu = rep(mu, p), Sigma = Sigma)
  colnames(X) <- paste0("X", 1:p)
  X
}

generate_beta_true <- function(p) {
  p1 <- round(p * 0.5); p2 <- round(p * 0.5); p3 <- p - p1 - p2
  beta_shuffled <- sample(c(runif(p1, 0, 1), runif(p2, 1, 2), rep(0, p3)))
  return(beta_shuffled)
}

generate_Y_fixed <- function(X, beta_true, R_squared, intercept = 10) {
  lin <- intercept + as.numeric(X %*% beta_true)
  vl  <- var(lin); if (vl <= .Machine$double.eps) vl <- .Machine$double.
      eps
  ve  <- vl * (1 - R_squared) / max(R_squared, 1e-8)
  lin + rnorm(nrow(X), 0, sqrt(ve))
}

generate_missing <- function(X, target_response_rate = 0.5) {
  n <- nrow(X)
  lin <- 0.2 * rowSums(X)
  trg <- (target_response_rate - 0.1) / 0.9
  trg <- min(max(trg, 0.01), 0.99)
  a <- as.numeric(quantile(lin, probs = 1 - trg))
  pr <- 0.1 + 0.9 * (1 / (1 + exp(-(lin - a))))
  rbinom(n, 1, pr)
}

# ------------------------------
# Math Helpers
# ------------------------------
ridge_closed_form <- function(X, y, lambda) {
  xbar <- colMeans(X); ybar <- mean(y)
  Xc <- sweep(X, 2, xbar, "-")
  yc <- y - ybar
  XtX <- crossprod(Xc)
  XtY <- crossprod(Xc, yc)
  A   <- XtX + diag(lambda, ncol(XtX))
  R   <- chol(A)
  beta <- backsolve(R, forwardsolve(t(R), XtY))
  intercept <- as.numeric(ybar - sum(xbar * beta))
```

```
  list(beta = as.numeric(beta), intercept = intercept, XtX = XtX, Xc = Xc,
       yc = yc)
}

align_intercept <- function(beta_vec, X_resp, y_resp) {
  ybar <- mean(y_resp)
  xbar <- colMeans(X_resp)
  as.numeric(ybar - sum(xbar * beta_vec))
}

debias_beta_k <- function(Rchol, beta_r, lambda, k) {
  solve_A <- function(v) backsolve(Rchol, forwardsolve(t(Rchol), v))
  v  <- solve_A(beta_r)
  s  <- lambda * v
  if (k >= 2) {
    for (j in 2:k) {
      v <- solve_A(v)
      s <- s + (lambda^j) * v
    }
  }
  beta_r + s
}

Ainv_power_vec <- function(Rchol, vec, m) {
  out <- vec
  for (i in 1:m) out <- backsolve(Rchol, forwardsolve(t(Rchol), out))
  out
}

Lk_apply_vec <- function(Rchol, vec, lambda, k) {
  out <- Ainv_power_vec(Rchol, vec, 1)
  if (k >= 1) {
    v <- out
    for (j in 1:k) {
      v <- backsolve(Rchol, forwardsolve(t(Rchol), v))
      out <- out + (lambda^j) * v
    }
  }
  out
}

cmse_hat_drk <- function(lambda, k, XtX, XtY, Xc, yc, Dx, nR, nNR) {
  n <- nR + nNR
  if (nNR <= 0) return(0)
  cfrac <- nNR / n
  p <- ncol(Xc)

  A <- XtX + diag(lambda, p)
  Rchol <- chol(A)

  beta_r <- backsolve(Rchol, forwardsolve(t(Rchol), XtY))

  # Squared bias (plug-in)
  Ainvk1_beta <- Ainv_power_vec(Rchol, beta_r, k + 1)
```

```r
  bdot <- as.numeric(crossprod(Dx, Ainvk1_beta))
  term_bias2 <- (lambda^(2*(k+1))) * (bdot^2)

  # Variance direction
  v_Dx <- Lk_apply_vec(Rchol, Dx, lambda, k)
  var_dir_core <- as.numeric(crossprod(v_Dx, XtX %*% v_Dx))

  # sigma^2
  denom <- max(1e-8, (nR - sum(XtX * chol2inv(Rchol))))
  sigma2_hat <- sum((yc - Xc %*% beta_r)^2) / denom

  # Covariance term
  sum_Xc <- colSums(Xc)
  term_cov <- (2 * sigma2_hat / nR) * sum(v_Dx * sum_Xc)

  (cfrac^2) * ( term_bias2 + sigma2_hat * var_dir_core + sigma2_hat * (1/nR
      + 1/nNR) + term_cov )
}

# --------------------------------
# Core Solver Wrappers
# --------------------------------

# 1. Solver for OLS
solve_ols <- function(X_curr, y_curr, R_curr) {
  X_resp <- X_curr[R_curr == 1, , drop=FALSE]
  y_resp <- y_curr[R_curr == 1]

  fit <- lm(y ~ ., data = data.frame(y = y_resp, X_resp))
  yhat <- predict(fit, newdata = data.frame(X_curr))

  y_filled <- y_curr
  y_filled[R_curr == 0] <- yhat[R_curr == 0]
  list(mu = mean(y_filled), y_filled = y_filled)
}

compute_cmse_eigen <- function(lambda_seq, eigen_dec, XtY, Xc, yc, Dx, nR,
    nNR, p, k_max = 0) {
  # Unpack Eigen decomposition: XtX = vectors %*% diag(values) %*% t(
     vectors)
  Q <- eigen_dec$vectors
  d <- eigen_dec$values # vector of eigenvalues

  # Q'XtY (Pre-calculate invariant part)
  Qt_XtY <- crossprod(Q, XtY) # p x 1

  # Pre-calculate Q'Dx (Pre-calculate invariant part)
  Qt_Dx  <- crossprod(Q, Dx)  # p x 1

  # Pre-calculate Xc Q (for sigma2 estimation)
  Xc_Q <- Xc %*% Q

  # Loop over lambdas using vectorization (vapply is faster than simple
     for loop)
```

```
  results <- vapply(lambda_seq, function(lam) {
    inv_d_lam <- 1 / (d + lam)

    # beta_r = Q * diag(inv_d_lam) * Qt_XtY
    # We work in rotated space to avoid matrix mults
    beta_rotated <- inv_d_lam * Qt_XtY

    # ------------------------------------------------
    # Best DR Logic (Loop over k)
    # ------------------------------------------------
    # We need to find best k for this specific lambda
    # Bias term: need (XtX + lam I)^-(k+1) beta_r
    # In rotated space: diag(inv_d_lam)^(k+1) * beta_rotated
    #                 = diag(inv_d_lam)^(k+2) * Qt_XtY

    vals_k <- numeric(k_max + 1)

    for (kk in 0:k_max) {
      # 1. Bias Squared
      # Rotated vector for bias calculation
      # power = (kk + 1) + 1 (because beta_rotated already has power 1) ->
          Wait.
      # Ainv_power_vec(beta, k+1) means A^-(k+1) * beta.
      # beta = A^-1 XtY. So total is A^-(k+2) XtY.

      factor_bias <- inv_d_lam^(kk + 1) # This applies to beta_rotated
      # bdot = Dx^T * A^-(k+1) * beta_r = (Q Qt_Dx)^T * Q * diag * Qt_beta
      #      = Qt_Dx^T * diag(factor) * beta_rotated
      bdot <- sum(Qt_Dx * factor_bias * beta_rotated)
      term_bias2 <- (lam^(2*(kk+1))) * (bdot^2)

      # 2. Variance Direction
      # v_Dx = Lk_apply(Dx). Lk = sum (lam^j A^-(j+1))
      # In rotated space, A^-(j+1) becomes inv_d_lam^(j+1)
      # So element-wise multiplier for Qt_Dx is: sum_{j=0}^k lam^j * inv_d
          _lam^(j+1)
      # Geometric series sum: u = lam * inv_d_lam.
      # Sum = inv_d_lam * (1 - u^(k+1)) / (1 - u).
      # But simpler to just loop since k is small (0-5)

      mult_vec <- inv_d_lam # j=0 term
      if (kk > 0) {
        term <- inv_d_lam
        for (j in 1:kk) {
          term <- term * (lam * inv_d_lam) # term becomes lam^j * inv_d_
              lam^(j+1)
          mult_vec <- mult_vec + term
        }
      }

      v_Dx_rot <- mult_vec * Qt_Dx # Rotated v_Dx

      # var_dir_core = v_Dx^T XtX v_Dx = v_Dx_rot^T diag(d) v_Dx_rot
      var_dir_core <- sum(v_Dx_rot^2 * d)
```

```
241
242       # 3. Sigma2
243       # beta = Q * beta_rotated
244       # Residuals: yc - Xc beta.
245       # Norm: ||yc||^2 - 2 yc^T Xc beta + beta^T XtX beta
246       # beta^T XtX beta = beta_rot^T diag(d) beta_rot
247       beta_XtX_beta <- sum(beta_rotated^2 * d)
248
249       # yc^T Xc beta = (Xc^T yc)^T beta = XtY^T beta = XtY^T Q beta_rot =
              Qt_XtY^T beta_rot
250       yc_Xc_beta <- sum(Qt_XtY * beta_rotated)
251
252       rss <- sum(yc^2) - 2 * yc_Xc_beta + beta_XtX_beta
253
254       # DoF: nR - trace(XtX A^-1) = nR - sum(d / (d+lam))
255       dof <- max(1e-8, nR - sum(d * inv_d_lam))
256       sigma2_hat <- rss / dof
257
258       # 4. Covariance
259       #Xc is centered, colSums(Xc) should be practically 0.
260       # So term_cov is 0. We can skip it or keep it for safety if
              centering isn't perfect.
261       # Assuming perfect centering:
262       term_cov <- 0
263
264       # CMSE Value
265       cfrac <- nNR / (nR + nNR)
266       val <- (cfrac^2) * (term_bias2 + sigma2_hat * var_dir_core + sigma2_
              hat*(1/nR + 1/nNR))
267       vals_k[kk+1] <- val
268     }
269
270     return(vals_k) # Returns vector of CMSE for k=0...k_max
271   }, numeric(k_max + 1))
272
273   return(results) # Matrix: (k_max+1) x length(lambda_seq)
274 }
275
276 # --------------------------------
277 # 2. Optimized Solvers
278 # --------------------------------
279
280 solve_ridge_cmse_eigen <- function(X_curr, y_curr, R_curr, lambda_grid) {
281   X_resp <- X_curr[R_curr == 1, , drop=FALSE]
282   y_resp <- y_curr[R_curr == 1]
283
284   nR <- sum(R_curr); nNR <- sum(R_curr == 0)
285   if (nNR == 0) return(list(mu = mean(y_resp), y_filled = y_curr))
286
287   # Centering
288   ybar_R <- mean(y_resp)
289   xbar_R <- colMeans(X_resp)
290   xbar_NR <- colMeans(X_curr[R_curr==0, , drop=FALSE])
291
```

```r
  Xc <- sweep(X_resp, 2, xbar_R, "-")
  yc <- y_resp - ybar_R
  Dx <- as.numeric(xbar_NR - xbar_R)

  XtX <- crossprod(Xc)
  XtY <- crossprod(Xc, yc)
  eigen_dec <- eigen(XtX, symmetric = TRUE) # p=10, super fast

  # CMSE (k=0)
  # returns 1 x n_lambda matrix
  cmse_vals <- compute_cmse_eigen(lambda_grid, eigen_dec, XtY, Xc, yc, Dx,
      nR, nNR, ncol(Xc), k_max=0)

  best_idx <- which.min(cmse_vals)
  lam_best <- lambda_grid[best_idx]

  # Compute final Beta
  # (XtX + lam I)^-1 XtY = Q (d+lam)^-1 Q' XtY
  # Q'XtY we calculated inside helper, but quick to redo or pass out
  Q <- eigen_dec$vectors
  d <- eigen_dec$values
  beta_ridge <- Q %*% ((1/(d + lam_best)) * crossprod(Q, XtY))

  b0 <- ybar_R - sum(xbar_R * beta_ridge)
  yhat <- as.numeric(b0 + X_curr %*% beta_ridge)
  y_filled <- y_curr
  y_filled[R_curr == 0] <- yhat[R_curr == 0]

  list(mu = mean(y_filled), y_filled = y_filled)
}

solve_best_dr_eigen <- function(X_curr, y_curr, R_curr, lambda_grid) {
  X_resp <- X_curr[R_curr == 1, , drop=FALSE]
  y_resp <- y_curr[R_curr == 1]

  nR <- sum(R_curr); nNR <- sum(R_curr == 0)
  if (nNR == 0) return(list(mu = mean(y_resp), y_filled = y_curr))

  ybar_R <- mean(y_resp)
  xbar_R <- colMeans(X_resp)
  xbar_NR <- colMeans(X_curr[R_curr==0, , drop=FALSE])

  Xc <- sweep(X_resp, 2, xbar_R, "-")
  yc <- y_resp - ybar_R
  Dx <- as.numeric(xbar_NR - xbar_R)

  XtX <- crossprod(Xc)
  XtY <- crossprod(Xc, yc)
  eigen_dec <- eigen(XtX, symmetric = TRUE)

  # Calculate CMSE for k=0..5 simultaneously
  # returns 6 x n_lambda matrix
  cmse_vals <- compute_cmse_eigen(lambda_grid, eigen_dec, XtY, Xc, yc, Dx,
      nR, nNR, ncol(Xc), k_max=5)
```

```
344
345   # Find global minimum
346   min_idx <- which.min(cmse_vals) # Linear index
347   best_k_idx <- (min_idx - 1) %% 6 + 1 # Row index (k+1)
348   best_lam_idx <- (min_idx - 1) %/% 6 + 1 # Col index
349
350   best_k <- best_k_idx - 1
351   best_lam <- lambda_grid[best_lam_idx]
352
353   # Compute Final Coefficient
354   Q <- eigen_dec$vectors
355   d <- eigen_dec$values
356   inv_d_lam <- 1 / (d + best_lam)
357   Qt_XtY <- crossprod(Q, XtY)
358
359   # Base Ridge (rotated)
360   beta_rot <- inv_d_lam * Qt_XtY
361
362   # Add Debiasing (rotated space)
363   # s = sum_{j=1}^k lam^j (d+lam)^-(j) * beta_rot
364   #   = beta_rot * sum_{j=1}^k (lam / (d+lam))^j
365
366   factor <- rep(1, length(d))
367   if (best_k > 0) {
368     ratio <- best_lam * inv_d_lam # lam / (d+lam)
369     term <- ratio
370     sum_ratios <- ratio
371     if (best_k > 1) {
372       for (j in 2:best_k) {
373         term <- term * ratio
374         sum_ratios <- sum_ratios + term
375       }
376     }
377     factor <- 1 + sum_ratios
378   }
379
380   beta_final_rot <- beta_rot * factor
381   beta_final <- Q %*% beta_final_rot
382
383   b0 <- ybar_R - sum(xbar_R * beta_final)
384   yhat <- as.numeric(b0 + X_curr %*% beta_final)
385   y_filled <- y_curr
386   y_filled[R_curr == 0] <- yhat[R_curr == 0]
387
388   list(mu = mean(y_filled), y_filled = y_filled)
389 }
390
391 solve_alasso_fast <- function(X_curr, y_curr, R_curr, gamma_val=1.0, eps_
      val=1e-6) {
392   X_resp <- X_curr[R_curr == 1, , drop=FALSE]
393   y_resp <- y_curr[R_curr == 1]
394
395   nfolds_fast <- 5
396
```

```r
  # Initial Ridge
  cvfit <- cv.glmnet(x = X_resp, y = y_resp, alpha = 0, nfolds = nfolds_fast,
                     standardize = FALSE, intercept = TRUE, parallel = FALSE)
  lam_cv <- cvfit$lambda.min
  beta_ridge <- as.numeric(coef(cvfit, s="lambda.min")[-1])

  w_alasso <- 1 / (abs(beta_ridge) + eps_val)^gamma_val

  # Adaptive Lasso
  cvfit_alasso <- cv.glmnet(x = X_resp, y = y_resp, alpha = 1, nfolds = nfolds_fast,
                            standardize = FALSE, intercept = TRUE,
                            penalty.factor = w_alasso, parallel = FALSE)

  coef_alasso <- as.matrix(coef(cvfit_alasso, s = "lambda.min"))
  beta_alasso <- as.numeric(coef_alasso[-1, , drop = TRUE])

  # glmnet handles intercept internally, retrieving it:
  b0_alasso <- coef_alasso[1, 1]

  yhat <- as.numeric(b0_alasso + X_curr %*% beta_alasso)
  y_filled <- y_curr
  y_filled[R_curr == 0] <- yhat[R_curr == 0]

  list(mu = mean(y_filled), y_filled = y_filled)
}


solve_post_alasso_fast <- function(X_curr, y_curr, R_curr, gamma_val=1.0, eps_val=1e-6) {
  X_resp <- X_curr[R_curr == 1, , drop=FALSE]
  y_resp <- y_curr[R_curr == 1]

  nfolds_fast <- 5

  cvfit <- cv.glmnet(x = X_resp, y = y_resp, alpha = 0, nfolds = nfolds_fast,
                     standardize = FALSE, intercept = TRUE, parallel = FALSE)
  beta_ridge <- as.numeric(coef(cvfit, s="lambda.min")[-1])
  w_alasso <- 1 / (abs(beta_ridge) + eps_val)^gamma_val


  cvfit_alasso <- cv.glmnet(x = X_resp, y = y_resp, alpha = 1, nfolds = nfolds_fast,
                            standardize = FALSE, intercept = TRUE,
                            penalty.factor = w_alasso, parallel = FALSE)

  coef_alasso <- as.matrix(coef(cvfit_alasso, s = "lambda.min"))
  beta_alasso <- as.numeric(coef_alasso[-1, , drop = TRUE])


```

```r
  selected_vars <- which(abs(beta_alasso) > 1e-8)


  if (length(selected_vars) == 0) {
    yhat <- rep(mean(y_resp), nrow(X_curr))
  } else {
    X_resp_sel <- X_resp[, selected_vars, drop=FALSE]
    fit_ols <- lm(y_resp ~ X_resp_sel)

    coef_ols <- coef(fit_ols)
    coef_ols[is.na(coef_ols)] <- 0

    X_curr_sel <- X_curr[, selected_vars, drop=FALSE]
    yhat <- as.numeric(coef_ols[1] + X_curr_sel %*% coef_ols[-1])
  }

  y_filled <- y_curr
  y_filled[R_curr == 0] <- yhat[R_curr == 0]

  list(mu = mean(y_filled), y_filled = y_filled)
}

# --------------------------------
# 3. Bootstrap Function
# --------------------------------

boot_variance_estimator_optimized <- function(X_s, y_filled_list, R_s, B,
    N_pop, lambda_grid, alasso_gamma, alasso_eps) {
  n <- nrow(X_s)
  k <- floor(N_pop / n)
  rem <- N_pop - n * k

  # Pre-calculate Pseudo-population indices
  idxs_U <- c(rep(1:n, k), sample(1:n, rem, replace = FALSE))

  # Truth theta*
  theta_stars <- lapply(y_filled_list, function(y) mean(y[idxs_U]))

  boot_sq_errs <- matrix(0, nrow = B, ncol = length(y_filled_list))
  colnames(boot_sq_errs) <- names(y_filled_list)

  # Vectorization Optimization: Generate all bootstrap indices at once
  # Matrix of indices: n x B
  # Each column is a SRSWOR from 1:N_pop
  # mapping back to 1:n via idxs_U

  # Since N_pop is large (2000) and n is small (200), we can just loop
      sampling
  # Generating indices is rarely the bottleneck compared to model fitting
  boot_idx_matrix <- matrix(0, nrow = n, ncol = B)
  for (b in 1:B) {
    boot_idx_matrix[, b] <- sample(1:N_pop, n, replace = FALSE)
  }

```

```
  for (b in 1:B) {
    idx_in_U <- boot_idx_matrix[, b]
    s_star_orig_idx <- idxs_U[idx_in_U]

    X_star <- X_s[s_star_orig_idx, ]
    R_star <- R_s[s_star_orig_idx]

    # Solve OLS
    if ("ols" %in% names(y_filled_list)) {
      y_star_truth <- y_filled_list[["ols"]][s_star_orig_idx]
      res <- solve_ols(X_star, y_star_truth, R_star)
      boot_sq_errs[b, "ols"] <- (res$mu - theta_stars[["ols"]])^2
    }

    # Solve Ridge CMSE (EIGEN ACCELERATED)
    if ("ridge_cmse" %in% names(y_filled_list)) {
      y_star_truth <- y_filled_list[["ridge_cmse"]][s_star_orig_idx]
      res <- solve_ridge_cmse_eigen(X_star, y_star_truth, R_star, lambda_
          grid)
      boot_sq_errs[b, "ridge_cmse"] <- (res$mu - theta_stars[["ridge_cmse"
          ]])^2
    }

    # Solve Alasso (FAST FOLDS)
    if ("alasso" %in% names(y_filled_list)) {
      y_star_truth <- y_filled_list[["alasso"]][s_star_orig_idx]
      res <- solve_alasso_fast(X_star, y_star_truth, R_star, alasso_gamma,
           alasso_eps)
      boot_sq_errs[b, "alasso"] <- (res$mu - theta_stars[["alasso"]])^2
    }

    # Solve Best DR (EIGEN ACCELERATED)
    if ("best_dr" %in% names(y_filled_list)) {
      y_star_truth <- y_filled_list[["best_dr"]][s_star_orig_idx]
      res <- solve_best_dr_eigen(X_star, y_star_truth, R_star, lambda_grid
          )
      boot_sq_errs[b, "best_dr"] <- (res$mu - theta_stars[["best_dr"]])^2
    }
    # Solve Post-Alasso
    if ("post_alasso" %in% names(y_filled_list)) {
      y_star_truth <- y_filled_list[["post_alasso"]][s_star_orig_idx]
      res <- solve_post_alasso_fast(X_star, y_star_truth, R_star, alasso_
          gamma, alasso_eps)
      boot_sq_errs[b, "post_alasso"] <- (res$mu - theta_stars[["post_
          alasso"]])^2
    }
  }

  colMeans(boot_sq_errs)
}

# --------------------------------
# Generate Global Data Once
# --------------------------------
```

```r
beta_true <- generate_beta_true(p)
X_pop     <- generate_X_population(N, p, rho)
X_pop_mm  <- as.matrix(X_pop)

cat("Predictors generated. Starting Simulation...\n")

# --------------------------------
# One Monte Carlo Iteration
# --------------------------------
one_mc <- function(iter) {

  set.seed(10000 + iter)

  # 1. Generate Y and R
  Y_pop   <- generate_Y_fixed(X_pop_mm, beta_true, R_squared)
  mu_true <- mean(Y_pop)

  s_idx <- sample.int(N, size = n, replace = FALSE)
  Xs <- X_pop_mm[s_idx, , drop = FALSE]
  ys <- Y_pop[s_idx]

  R <- generate_missing(Xs, 0.5)
  if (sum(R) < p + 2) R <- generate_missing(Xs, 0.5)

  # 2. Run Point Estimators (Get y_filled for each)
  res_ols    <- solve_ols(Xs, ys, R)
  res_ridge  <- solve_ridge_cmse_eigen(Xs, ys, R, lambda_grid_cmse)
  res_alasso <- solve_alasso_fast(Xs, ys, R, alasso_gamma, alasso_eps)
  res_post_alasso <- solve_post_alasso_fast(Xs, ys, R, alasso_gamma,
     alasso_eps)
  res_best   <- solve_best_dr_eigen(Xs, ys, R, lambda_grid_cmse)

  # Pack results
  y_filled_list <- list(
    ols = res_ols$y_filled,
    ridge_cmse = res_ridge$y_filled,
    alasso = res_alasso$y_filled,
    post_alasso = res_post_alasso$y_filled,
    best_dr = res_best$y_filled
  )

  set.seed(20000 + iter)
  # 3. Optimized Bootstrap (All in One)
  boot_mses <- boot_variance_estimator_optimized(
    X_s = Xs,
    y_filled_list = y_filled_list,
    R_s = R,
    B = B_boot,
    N_pop = N,
    lambda_grid = lambda_grid_cmse,
    alasso_gamma = alasso_gamma,
    alasso_eps = alasso_eps
  )

```

```r
  # Return Results
  c(
    mu_imp_ols    = res_ols$mu,
    mu_imp_ridge  = res_ridge$mu,
    mu_imp_alasso = res_alasso$mu,
    mu_imp_post_alasso = res_post_alasso$mu,
    mu_imp_best   = res_best$mu,
    mu_true       = mu_true,

    Boot_MSE_OLS    = boot_mses["ols"],
    Boot_MSE_Ridge  = boot_mses["ridge_cmse"],
    Boot_MSE_Alasso = boot_mses["alasso"],
    Boot_MSE_Post_Alasso = boot_mses["post_alasso"],
    Boot_MSE_Best   = boot_mses["best_dr"]
  )
}

# --------------------------------
# Parallel Execution
# --------------------------------
n_cores <- max(1L, min(detectCores(logical = FALSE) - 1L, 30L))
cl <- makeCluster(n_cores)
clusterSetRNGStream(cl, 1)

clusterEvalQ(cl, {
  library(glmnet); library(MASS); library(Matrix)
})

clusterExport(cl, varlist = ls(), envir = environment())

res_list <- pblapply(1:M, function(i) one_mc(i), cl = cl)
stopCluster(cl)

res_mat <- do.call(rbind, res_list)

# --------------------------------
# Summary Output
# --------------------------------
# 1. Extract Vectors
mu_true_vec <- res_mat[, "mu_true"]
mu_ols_vec <- res_mat[, "mu_imp_ols"]
mu_ridge_vec <- res_mat[, "mu_imp_ridge"]
mu_alasso_vec <- res_mat[, "mu_imp_alasso"]
mu_post_alasso_vec <- res_mat[, "mu_imp_post_alasso"]
mu_best_vec <- res_mat[, "mu_imp_best"]

boot_mse_ols_vec <- res_mat[, "Boot_MSE_OLS.ols"]
boot_mse_ridge_vec <- res_mat[, "Boot_MSE_Ridge.ridge_cmse"]
boot_mse_alasso_vec <- res_mat[, "Boot_MSE_Alasso.alasso"]
boot_mse_post_alasso_vec <- res_mat[, "Boot_MSE_Post_Alasso.post_alasso"]
boot_mse_best_vec <- res_mat[, "Boot_MSE_Best.best_dr"]

# 2. Calculate True Monte Carlo MSE
mc_mse_ols    <- mean((mu_ols_vec - mu_true_vec)^2)
```

```r
mc_mse_ridge  <- mean((mu_ridge_vec - mu_true_vec)^2)
mc_mse_alasso <- mean((mu_alasso_vec - mu_true_vec)^2)
mc_mse_post_alasso <- mean((mu_post_alasso_vec - mu_true_vec)^2)
mc_mse_best   <- mean((mu_best_vec - mu_true_vec)^2)

# 3. Calculate Average Bootstrap MSE
avg_boot_mse_ols    <- mean(boot_mse_ols_vec)
avg_boot_mse_ridge  <- mean(boot_mse_ridge_vec)
avg_boot_mse_alasso <- mean(boot_mse_alasso_vec)
avg_boot_mse_best   <- mean(boot_mse_best_vec)
avg_boot_mse_post_alasso <- mean(boot_mse_post_alasso_vec)

# 4. Construct Comparison Table
df_res <- data.frame(
  Estimator = c("OLS", "Ridge (CMSE)", "Adaptive Lasso", "Post-Adaptive
     Lasso", "Best Debiased Ridge"),

  MonteCarlo_MSE = c(mc_mse_ols, mc_mse_ridge, mc_mse_alasso, mc_mse_post_
     alasso, mc_mse_best),

  Avg_Bootstrap_MSE = c(avg_boot_mse_ols, avg_boot_mse_ridge, avg_boot_mse
     _alasso,avg_boot_mse_post_alasso, avg_boot_mse_best)
)

df_res$Rel_Bias <- (df_res$Avg_Bootstrap_MSE - df_res$MonteCarlo_MSE) / df
    _res$MonteCarlo_MSE
df_res$Rel_Bias_Pct <- sprintf("%.2f%%", df_res$Rel_Bias * 100)

cat("\n============================================================\n")
cat(" Bootstrap MSE Estimator Results (B =", B_boot, ")\n")
cat("============================================================\n")
print(df_res)
```