

FEDERAL UNIVERSITY OF MINAS GERAIS

**Institute of Exact Sciences
Specialization Program in Statistics**

Gustavo Macedo Miranda

Comparative Analysis of Variable Selection Techniques

Belo Horizonte

2024

Gustavo Macedo Miranda

Comparative Analysis of Variable Selection Techniques

Specialization monograph presented to the Institute of Exact Sciences of the Federal University of Minas Gerais, as a partial requirement for obtaining the title of Specialist in Statistics.

Advisor: Professor Doctor Marcos Oliveira Prates

Belo Horizonte

2024

	Miranda, Gustavo Macedo.
M672c	Comparative analysis of variable selection techniques [recurso eletrônico] / Gustavo Macedo Miranda. –2024. 1 recurso online (55 f. il.) : pdf. Orientador: Marcos Oliveira Prates. Monografia(especialização) - Universidade Federal de Minas Gerais, instituto de Ciências Exatas, Departamento de Estatística. Referências: f.55. 1. Estatística. 2. Análise de regressão. 3. Variedades (Matemática). 4. LASSO (Estatística). 5. Aprendizado do Computador. I. Prates, Marcos Oliveira. II. Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Estatística. III. Título. CDU 519.2(043)

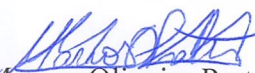


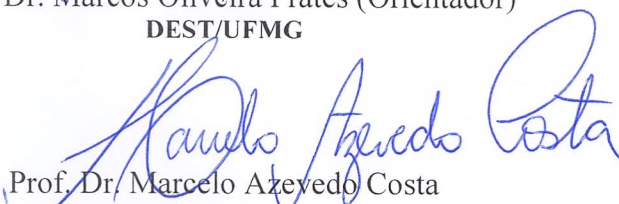
Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Estatística
Programa de Pós-Graduação / Especialização
Av. Pres. Antônio Carlos, 6627 - Pampulha
31270-901 – Belo Horizonte – MG

E-mail: pgest@ufmg.br
Tel: 3409-5923 – FAX: 3409-5924

ATA DO 332º. TRABALHO DE FIM DE CURSO DE ESPECIALIZAÇÃO EM ESTATÍSTICA DE GUSTAVO MACEDO MIRANDA.

Aos vinte e cinco dias do mês de junho de 2024, às 14:30 horas, reuniram-se os professores abaixo relacionados na sala 2076, no prédio do Instituto de Ciências Exatas, formando a Comissão Examinadora homologada pela Comissão do Curso de Especialização em Estatística, para julgar a apresentação do trabalho de fim de curso do aluno **Gustavo Macedo Miranda**, intitulado: “Comparative Analysis of Variable Selection Techniques”, como requisito para obtenção do Grau de Especialista em Estatística. Abrindo a sessão, o Presidente da Comissão, Professor Marcos Oliveira Prates – Orientador, após dar conhecimento aos presentes do teor das normas regulamentares, passou a palavra ao candidato para apresentação de seu trabalho. Seguiu-se a arguição pelos examinadores com a respectiva defesa do candidato. Após a defesa, os membros da banca examinadora reuniram-se sem a presença do candidato e do público, para julgamento e expedição do resultado final. Foi atribuída a seguinte indicação: o candidato foi considerado Aprovado condicional às modificações sugeridas pela banca examinadora no prazo de 30 dias a partir da data de hoje por unanimidade. O resultado final foi comunicado publicamente à candidata pelo Presidente da Comissão. Nada mais havendo a tratar, o Presidente encerrou a reunião e lavrou a presente Ata, que será assinada por todos os membros participantes da banca examinadora. Belo Horizonte, 25 de junho de 2024.


Prof. Dr. Marcos Oliveira Prates (Orientador)
DEST/UFMG


Prof. Dr. Marcelo Azevedo Costa
Escola de Engenharia/UFMG

Acknowledgements

To God, for being with me in good times and especially in challenging ones, giving me the strength needed to overcome this journey.

To my parents, Marta and Nelson, for the teachings and values instilled in me, and to my siblings, Danilo and Raquel, for their love and companionship.

To my advisor, Professor Marcos, for the support and knowledge shared to help me complete this work, and for challenging me to do more and better.

To the university community for providing the opportunity to access a quality education.

Resumo

Este estudo avalia alguns métodos de seleção de variáveis através de simulações em 19 cenários distintos, variando em tamanho da amostra n , número de variáveis p , níveis de significância, forças dos coeficientes beta e tipos de variáveis (discretas e contínuas). Métodos tradicionais como Backward e Stepwise demonstraram desempenho robusto em diferentes valores de n , enquanto abordagens mais recentes como SSGL mostraram maior eficácia em cenários com maior p ou coeficientes beta mais fortes. Todos os métodos apresentaram alta performance na maioria dos cenários devido à sua capacidade de identificar variáveis relevantes, ou seja, através da métrica de sensibilidade.

Além disso, a análise de eficiência computacional revelou que o LASSO exigiu tempo de processamento pequeno em todos os cenários em comparação com os métodos Backward e Stepwise. Já o SSGL, apresentou tempos de processamento variáveis influenciados pela força beta, n e p . Tanto os métodos Backward quanto Stepwise exibiram comportamento computacional semelhante, com Stepwise consistentemente requerendo mais tempo à medida que p aumentava. A aplicação a dados do mundo real sublinhou a utilidade prática desses métodos, apoiando sua seleção com base nas características específicas dos dados e nos objetivos de desempenho. Esta pesquisa contribui com insights cruciais para a otimização de estratégias de seleção de variáveis em modelagem estatística e análises preditivas.

Palavras-chave: *variável; sensibilidade; lasso; seleção de variáveis.*

Abstract

This study rigorously assesses various methods for selecting variables through simulation across 19 distinct scenarios, varying in sample size n , number of variables p , significance levels, beta coefficient strengths, and variable types (discrete and continuous). Traditional methods like Backward and Stepwise consistently demonstrated robust performance across different n values, whereas newer approaches such as SSGL showed enhanced efficacy in scenarios with larger p or stronger beta coefficients. All methods demonstrated high performance in most scenarios due to their ability to identify relevant variables, as evidenced by the sensitivity metric.

Moreover, computational efficiency analysis revealed that LASSO required minimal processing time across all scenarios, contrasting with SSGL, which showed variable computation times influenced by beta strength, n , and p . Both Backward and Stepwise methods exhibited similar computational behavior, with Stepwise consistently requiring more time as p increased. Application to real-world data underscored the practical utility of these methods, supporting their selection based on specific data characteristics and performance objectives. This research contributes crucial insights for optimizing variable selection strategies in statistical modeling and predictive analytics.

Keywords: *variable; sensitivity; lasso; selection methods.*

List of Figures

Figure 1 – Mean time per scenario for small values of n and p	22
Figure 2 – Mean Time per Scenario across Varying Beta Strengths	24
Figure 3 – Mean Time per Scenario across Varying Beta Strengths and Large n Values	26
Figure 4 – Mean Time per Scenario across Varying Beta Strengths and Large p Values	27
Figure 5 – Mean Time per Scenario: Percentage of Significant Variables	29
Figure 6 – Mean Time per Scenario: Variation Across Types of Variables	30
Figure 7 – Mean time per scenario: Increase in n Value	32
Figure 8 – Mean time per scenario: Increase in p Value	34
Figure 9 – Correlation matrix - anthropometric data	37
Figure 10 – Normality test - anthropometric data	52
Figure 11 – Variance test - anthropometric data	53

List of Tables

Table 1 – Scenarios and Parameters for Analysis	20
Table 2 – Parameter Settings for Scenarios 1, 2, and 3	22
Table 3 – Performance Metrics for Scenarios 1, 2, and 3	23
Table 4 – Parameter Settings for Scenarios 4, 5, and 6	23
Table 5 – Performance Metrics for Scenarios 1, 2, and 3	25
Table 6 – Parameter Settings for Scenarios 10, 11, and 12	25
Table 7 – Performance Metrics for Scenarios 10, 11, and 12	26
Table 8 – Parameter Settings for Scenarios 7, 8, and 9	26
Table 9 – Performance Metrics for Scenarios 7, 8, and 9	28
Table 10 – Parameter Settings for Scenarios 13, 14, and 15	28
Table 11 – Performance Metrics for Scenarios 13, 14, and 15	29
Table 12 – Parameter Settings for Scenarios 14, 16, and 17	30
Table 13 – Performance Metrics for Scenarios 14, 16, and 17	31
Table 14 – Parameter Settings for Scenarios 5, 11, and 18	32
Table 15 – Performance Metrics for Scenarios 5, 11, and 18	33
Table 16 – Parameter Settings for Scenarios 5, 8, and 19	34
Table 17 – Performance Metrics for Scenarios 5, 8, and 19	35
Table 18 – OLS Regression Results - Model1	39
Table 19 – Regression coefficients - Model1	39
Table 20 – OLS Regression Results - Model2	41
Table 21 – Regression Coefficients - Model2	41
Table 22 – Output Backward - AIC	42
Table 23 – Output Backward - BIC	43
Table 24 – Output Stepwise - AIC	45
Table 25 – Output Stepwise - BIC	45
Table 26 – Exhaustive search	46
Table 27 – Model Indicators	47
Table 28 – Coefficients of the Lasso Model	48
Table 29 – Coefficients of the Lasso Model	49
Table 30 – Variable Selection by Different Methods	50

Contents

1	INTRODUCTION	11
2	METHODOLOGY	12
3	LITERATURE REVIEW	13
4	RESULTS AND DISCUSSION	16
4.1	The methods	16
4.2	The simulation and the data	18
4.3	Output analysis	20
4.3.1	Case 1: Beta Variation for Small Values of n and p	22
4.3.2	Case 2: Beta Variation for Large Values of n and p	23
4.3.3	Case 3: Beta Variation for Increasing n value	24
4.3.4	Case 4: Beta Variation for Increasing p value	25
4.3.5	Case 5: Variation of the Percentage of Significant Variables	27
4.3.6	Case 6: Variation in the Type of Variables	30
4.3.7	Case 7: Increase in n Value	32
4.3.8	Case 8: Increase in p Value	34
4.4	Real example: Anthropometric Data	36
4.4.1	Initial analysis	36
4.4.2	Backward Stepwise Selection	41
4.4.3	Forward Stepwise Selection	43
4.4.4	Stepwise Selection	44
4.4.5	Exhaustive search	45
4.4.6	Lasso	47
4.4.7	SSGL	48
4.4.8	Final Remarks	50
5	CONCLUSION	55
6	BIBLIOGRAPHY	56

1 INTRODUCTION

A field as critical to scientific and industrial inquiry as Statistics continually evolves in response to emerging challenges. Originating from modest-scale agricultural and industrial experiments, these challenges have expanded exponentially with advancements in computing technology and the digital era. The current landscape across various disciplines is characterized by the generation of vast amounts of data, necessitating statisticians to decipher intricate patterns and trends within this expansive information reservoir. This fundamental process, known as learning from data, remains pivotal in extracting meaningful insights from datasets (Hastie et al., 2009). The continuous growth of data, aided by widespread personal computing, presents a challenge in managing and extracting meaningful insights from an overwhelming amount of information (Witten et al., 2011).

The primary objective of this study is to rigorously evaluate and compare the performance of various variable selection methods. Initially, a comprehensive simulation was conducted to assess the accuracy, sensitivity, specificity, and F1 score of established methods such as Backward, Stepwise, Lasso (Tibshirani, 1996), and SSGL - Spike-and-Slab Group Lasso (Ročková and George, 2018), across 19 distinct scenarios. These scenarios were meticulously designed with variations in sample size n , number of predictors p , proportion of significant variables, strength of beta coefficients, and types of variables (discrete and continuous).

Following the simulation, these methods were further applied to real-world example to observe their behavior and validate the findings from simulated data. Prior to delving into the variable selection process, an exploratory data analysis was performed to glean insights into the dataset's characteristics. This preliminary analysis included examining variable correlations, identifying potential outliers, and assessing data distributions. A robust understanding of these aspects is crucial for making informed decisions during variable selection and ensuring the integrity of subsequent regression models.

The analysis revealed nuanced performance trends: traditional methods like Backward and Stepwise generally excelled in scenarios with smaller or larger sample sizes of n . In contrast, newer methods such as Lasso and SSGL demonstrated superior performance when faced with larger numbers of predictors or stronger beta coefficients. Interestingly, Backward and Stepwise methods consistently produced comparable outcomes across different scenarios, with sensitivity emerging as a prominent metric for accurately identifying relevant variables. This study not only identifies optimal methods tailored to specific data characteristics and performance criteria but also provides practical insights for enhancing statistical modeling and predictive analytics in diverse application domains.

2 METHODOLOGY

This study adopts a quantitative approach employing simulations to assess and compare the performance of various variable selection methods. Simulated data is generated to encompass diverse scenarios, manipulating parameters such as sample size n , number of variables p , percentage of significant variables, beta coefficient strength, and variable type (discrete and continuous).

Central to the methodology is the utilization of R (R Core Team, 2024) and Google Colab (Google Colaboratory, 2024) as primary software tools. R is utilized for implementing variable selection methods such as Backward, forward, Stepwise, exhaustive, and Lasso, leveraging dedicated functions and packages. Additionally, Python (Python Software Foundation, 2024) libraries in Google Colab are employed for data preprocessing and exploratory data analysis tasks, ensuring a comprehensive approach to model development and evaluation.

Simulations are conducted using these computational tools, focusing on statistical analysis and modeling. Google Sheets is employed for data analysis and result visualization, capitalizing on its analytical capabilities and collaborative features. The computational environment is based on a Windows 11 operating system, running on an Intel Core i7 processor, optimized for the intensive calculations required by simulation and analysis tasks.

This methodology ensures a rigorous and systematic evaluation of variable selection methods across simulated scenarios. By leveraging advanced computational tools and platforms, the study aims to provide insights into the efficacy of these methods under varying data conditions. The findings contribute to enhancing decision-making processes in statistical modeling and predictive analytics, addressing specific performance requirements based on data characteristics.

3 LITERATURE REVIEW

In subset selection, variables are retained or eliminated from the model based on different strategies. Hastie et al. (2009) describe best subset regression, which aims to find the subset of variables of size k that minimizes the residual sum of squares. This method employs an efficient algorithm, such as the leaps and bounds procedure, particularly suitable for handling large p values. Conversely, forward Stepwise selection begins with the intercept and iteratively adds the predictor that most enhances the model fit. It produces a sequence of nested models indexed by k , the subset size, which must be determined. On the other hand, Backward Stepwise selection starts with the full model and sequentially removes the predictor with the least impact on the fit. This method can only be utilized when the number of observations n exceeds the number of predictors p , whereas forward Stepwise selection can be employed regardless of the n to p ratio.

These methods are subsets of the broader subset selection approach, wherein variables are systematically included or excluded from the model based on specific criteria. The choice of method often depends on computational feasibility, statistical considerations, and the desired balance between bias and variance (Hastie et al., 2009).

Tibshirani (1996) introduced the LASSO (Least Absolute Shrinkage and Selection Operator) method for shrinkage and variable selection in regression and generalized regression problems. Unlike traditional subset selection approaches, the LASSO employs a continuous shrinking operation that can precisely set coefficients to zero, facilitating automatic variable selection.

The LASSO minimizes the following objective function:

$$\min_{\beta} \left\{ \frac{1}{2n} \sum_{i=1}^n (y_i - x_i^T \beta)^2 + \lambda \sum_{j=1}^p |\beta_j| \right\}$$

Here, λ is the penalty parameter that controls the amount of regularization. A larger λ increases the shrinkage, potentially driving more coefficients β_j to zero, thus facilitating variable selection (Tibshirani, 1996).

Through empirical evaluation, Tibshirani demonstrates the competitiveness of the LASSO against subset selection and ridge regression across various scenarios. Specifically, in scenarios with a small number of large effects, subset selection outperforms, followed by the LASSO and ridge regression. Conversely, when dealing with a small to moderate number of moderate-sized effects, the LASSO excels, followed by ridge regression and then subset selection. In cases involving a large number of small effects, ridge regression demonstrates superior performance, followed by the LASSO and subset selection. Tibshirani's work establishes the LASSO as a

versatile and effective tool for regression analysis, offering valuable insights into its relative strengths and weaknesses compared to other regularization methods.

Ročková and George (2018) present the Spike-and-Slab LASSO method, which introduces a novel approach to variable selection in high-dimensional regression scenarios where $p > n$. This method utilizes a penalized likelihood framework with continuous spike-and-slab priors, offering insights into the treatment of such priors and their application in self-adaptive and automatically tuned high-dimensional regression models. Unlike traditional LASSO methods that utilize single Laplace priors with increasing penalties, the Spike-and-Slab LASSO employs a sequence of Laplace mixtures with an escalating spike penalty λ_0 , while maintaining a fixed small constant λ_1 .

The Spike-and-Slab LASSO can be formalized as follows:

$$\hat{\beta} = \arg \min_{\beta} \left\{ \frac{1}{2} \|y - X\beta\|_2^2 + \sum_{j=1}^p [\lambda_1(1 - \gamma_j)|\beta_j| + \lambda_0\gamma_j|\beta_j|] \right\}$$

where λ_0 and λ_1 are regularization parameters corresponding to the "spike" and "slab" components, respectively, and $\gamma_j \in [0, 1]$ is a binary indicator determining whether the j -th coefficient β_j is subject to the "spike" (if $\gamma_j = 1$) or the "slab" (if $\gamma_j = 0$) penalty.

This formulation enables the model to adaptively select relevant variables by shrinking some coefficients towards zero (spike) while allowing others to remain relatively large (slab), making it particularly well-suited for high-dimensional settings.

The Spike-and-Slab LASSO employs a path-following approach to dynamically explore the posterior distribution. By considering a sequence of increasing spike penalty parameters λ_0 , the method gradually filters out noisy coefficients while preserving meaningful ones. This strategy, described in detail by Ročková and George (2018), starts with a small λ_1 to ensure that the posterior distribution is not overly spiky. As the penalty parameter increases, the method effectively removes irrelevant coefficients, akin to the traditional LASSO solution path. However, the inclusion of a slab component prevents large coefficients from being overly penalized, allowing them to remain in the model. This iterative process ultimately facilitates efficient variable selection while maintaining model interpretability and predictive accuracy.

In their recent work, Bai et al. (2020) introduced the Spike-and-Slab Group Lasso (SSGL) method tailored for Bayesian grouped regression and sparse generalized additive models. Under the SSGL prior, the global posterior mode exhibits exact sparsity, effectively identifying and excluding insignificant groups of coefficients. By extending SSGL to sparse generalized additive models through the nonparametric Spike-and-Slab Lasso (NPSSL), the authors offer a flexible framework capable of adapting to a wide range of modeling scenarios. Their contributions include proposing a novel group spike-and-slab prior, developing an efficient optimization algorithm for mode estimation, and demonstrating the applicability of de-biasing techniques for valid inference

on regression coefficients, thereby providing a comprehensive approach for variable selection and estimation in both parametric and nonparametric settings. Additionally, SSGL introduces an adaptive regularization parameter, which adjusts shrinkage levels individually for each group, contrasting with the group lasso where the same shrinkage is uniformly applied across all groups.

The Spike-and-Slab Group Lasso (SSGL) method introduced by Bai et al. (2020) is an extension of the Spike-and-Slab Lasso (SSL) for grouped regression. The mathematical formulation of SSGL can be represented as:

$$\hat{\beta} = \arg \min_{\beta} \left\{ \frac{1}{2} \|y - X\beta\|_2^2 + \sum_{g=1}^G [\lambda_1(1 - \gamma_g) \|\beta_g\|_2 + \lambda_0 \gamma_g \|\beta_g\|_2] \right\}$$

Here:

β_g represents the coefficient vector associated with group g .

$\|\beta_g\|_2$ is the ℓ_2 norm of the coefficients in group g , used to penalize entire groups of variables.

$\gamma_g \in [0, 1]$ is a binary indicator that determines whether group g is penalized by the "spike" component ($\gamma_g = 1$) or the "slab" component ($\gamma_g = 0$).

λ_0 and λ_1 are the regularization parameters corresponding to the "spike" and "slab" components, respectively.

While the Spike-and-Slab Lasso (SSL) deals with individual coefficients, the SSGL applies to groups of coefficients, which is useful in scenarios where variables are naturally grouped.

Furthermore, Bai et al. (2020) suggest maintaining a consistent, small value for the slab hyperparameter λ_1 , as they find that its choice does not significantly impact the results. This parameter regulates the variance of the slab component of the prior, necessitating a sufficiently large variance to prevent excessive shrinkage of important covariates. For their default implementation, they advocate for setting $\lambda_1=1$, which imposes minimal shrinkage on significant coefficient groups, allowing them to remain unaffected by the spike. Meanwhile, they recommend selecting the spike parameter λ_0 from an increasing ladder of values, suggesting λ_0 [1, 2, ..., 100] to encompass a spectrum from minimal to very strong penalization (Bai et al., 2020).

4 RESULTS AND DISCUSSION

This chapter can be divided into two parts. The first part will discuss the concept and analysis of a simulation comparing various selection methods. In the second part, the analysis and description of one real databases will be presented, followed by the development of the model and residual analysis. Additionally, different methods for variable selection will be applied to the example.

4.1 The methods

Regarding the simulation, data was generated based on specific criteria described subsequently, and the methods tested include Backward, Stepwise, lasso, and ssgl. The analysis focuses on directly comparing these methods:

- **Backward Selection:** In Backward selection, the model starts with all predictors and iteratively removes the least significant predictor until a stopping criterion is met. It is a Stepwise regression approach that begins with the full model and progressively eliminates variables (Hastie, Tibshirani e Friedman, 2008).
- **Stepwise Selection:** Stepwise selection is a method that adds or removes predictors at each Step based on predefined criteria. It can be forward, Backward, or both; in this case, both are employed (Hastie, Tibshirani e Friedman, 2008).
- **Lasso (Least Absolute Shrinkage and Selection Operator):** Lasso is a regularization technique that penalizes the absolute size of the coefficients, leading many of them to be exactly zero. This effectively performs variable selection by shrinking the coefficients of less important variables to zero, thus automatically selecting a subset of features (Hastie, Tibshirani e Friedman, 2008).
- **SSGL (Sparse Group Lasso):** SSGL is an extension of Lasso that applies variable selection to groups of variables. It encourages sparsity both within and between groups of predictors, making it suitable for problems where predictors can be naturally grouped together. SSGL operates similarly to Lasso, but with the distinction that SSGL utilizes two λ parameters, whereas Lasso utilizes only one (Ročková and George, 2018).

When considering the selection methods, it's important to explore their specifications and R arguments. In Backward and Stepwise approaches, the Step function is utilized with the `direction = 'Backward'` and `direction = 'both'` arguments, respectively. The criterion for variable

selection in these methods is based on the significance of the variables as determined by the F-test.

Moving on to Lasso, it's executed with the $\alpha = 1$ argument, and λ is chosen through cross-validation using the `glmnet` function. Similarly, for SSGL, the `glmnet` function is employed, with λ_1 set at a fixed value of 1 and λ_0 determined through cross-validation within the range from 1 to 100, following the guidelines provided by Bai et al. (2020). The code used to run each method is shown in Listing 4.1.

Listing 4.1 – R Code for Selection Methods

```

1 R input:
2
3 ##Backward
4   model <- lm(y ~ ., data = as.data.frame(x))
5   metod_back <- step(model, direction = 'Backward', test = 'F')
6
7 ##Stepwise
8   metod_step <- step(model, direction = 'both', test = 'F')
9
10 ##Lasso
11   cv <- cv.glmnet(as.matrix(x), y, alpha = 1)
12   best_lambda <- cv$lambda.min
13   metod_lasso <- glmnet(x, y, alpha = 1, lambda = best_lambda)
14
15 ##SSGL
16   groups <- 1:ncol(x)
17   n <- nrow(x)
18   train_index <- sample(1:n, round(0.8*n))
19   test_index <- setdiff(1:n, train_index)
20
21   # training
22   X_train <- x[train_index, ]
23   Y_train <- y[train_index]
24
25   # test
26   X_test <- x[test_index, ]
27   Y_test <- y[test_index]
28
29   # cross validation
30   lambda1 <- 1
31   lambda0seq <- seq(1, 100, length = 100)
32
33   error_min <- Inf
34   best_lambda <- NULL
35
36   for (lambda0 in lambda0seq) {
37     modSSGL <- SSGL(Y = Y_train, X = X_train, lambda1 = lambda1, lambda0 =

```

```

        lambda0, groups = groups,
38         a = 1, b = ncol(X_train), M = 10, error = 0.001)
39     error <- mean((Y_test - X_test %*% modSSGL$beta)^2)
40
41     if (error < error_min) {
42         error_min <- error
43         best_lambda <- c(lambda1, lambda0)
44     }
45 }
46
47 metod_ssgl <- SSGl(Y = y, X = x, lambda1 = best_lambda[1], lambda0 = best
    _lambda[2], groups = groups, a = 1, b = ncol(x), M = 10, error =
    0.001)

```

4.2 The simulation and the data

To evaluate the methods via simulation, nineteen scenarios were constructed, where the code was run in a loop one hundred times to capture the variability of data. For each of the one hundred iterations, the four methods (Backward, step, Lasso, and SSGl) were executed once.

The simulation manipulated and inputted five parameters to evaluate and analyze each scenario: the number of observations (n), the number of predictor variables (p), the proportion of significant response variables (where β is greater than zero), the range of variation of β strength values (ranging from 0 to 0.2, from 0.5 to 1 or from 1 to 2), and the nature of the variables (continuous, discrete or a mix of continuous and discrete). Additionally, the time spent running the code for each scenario and the mean time spent running each method were recorded. This process unfolded across various scenarios, with a minimum of one hundred random y vectors generated and tested for each to capture the variability in the data. Each of these was called an 'iteration'.

The data were generated based on an x -matrix with dimensions specified by random values normally distributed. Subsequently, a β vector was created with random values, strategically placing non-zero values in certain positions. For instance, half of the vector contained random values while the other half consisted of zeros. This strategy aimed to evaluate the effectiveness of variable selection methods in identifying significant variables. Next, the y vector was formulated based on the x -matrix and β vector, with the addition of random error.

It is important to note that the analysis presupposes that the significant variables, where β is greater than zero, are crucial to the model and contribute to generating the best model. In other words, the best selection method is the one that can select all significant variables and dismiss those where β is zero. Subsequently, a description of the data, specifications about the scenarios, and an analysis of each will be provided.

To clarify the execution path for all scenarios, the following step-by-step approach was

employed:

- Firstly, the number of observations (n) was defined, ranging from small values like 20 to larger ones like 5000;
- Subsequently, the number of variables (p) was determined, ranging from 10 to 300, with one of the scenarios having p larger than n ;
- The proportion of significant variables was then specified at 20%, 50%, or 70%;
- Then the magnitude of β values was defined, ranging from 0 to 0.2, from 0.5 to 1, or from 1 to 2, to assess its influence on selection methods;
- Lastly, the distribution of variables was determined as either continuous, discrete, or a mix of continuous and discrete.

This systematic approach aimed to comprehensively evaluate the performance of variable selection methods across a range of scenarios, considering various factors that could influence their effectiveness. The scenarios are better illustrated in Table 1, where it is possible to observe, following the columns, the scenario number, n , p , the percentage of variables where β is greater than zero, the range of β , and the type of variable considered for each one.

To better evaluate each of the indicators, some of the scenarios were grouped logically. For example, scenarios 1, 2, and 3 contain the same n and p values, the same percentage of variables where β is greater than zero, and the same type of variables, but different β ranges. In this case, the objective was to evaluate the influence of the variance of β significance when n and p have small values (Table 1). This logic was applied to other scenarios, and each will be better explored in the next section with detailed analysis.

Table 1 – Scenarios and Parameters for Analysis

All Scenario	n	p	% variables $\beta > 0$	Beta range	Type of var
Scenario 1	20	10	70%	0 - 0.2	Continuous
Scenario 2	20	10	70%	0.5 - 1	Continuous
Scenario 3	20	10	70%	1 - 2	Continuous
Scenario 4	500	100	50%	0 - 0.2	Continuous
Scenario 5	500	100	50%	0.5 - 1	Continuous
Scenario 6	500	100	50%	1 - 2	Continuous
Scenario 7	500	200	50%	0 - 0.2	Continuous
Scenario 8	500	200	50%	0.5 - 1	Continuous
Scenario 9	500	200	50%	1 - 2	Continuous
Scenario 10	1000	100	50%	0 - 0.2	Continuous
Scenario 11	1000	100	50%	0.5 - 1	Continuous
Scenario 12	1000	100	50%	1 - 2	Continuous
Scenario 13	200	50	20%	0.5 - 1	Continuous
Scenario 14	200	50	50%	0.5 - 1	Continuous
Scenario 15	200	50	70%	0.5 - 1	Continuous
Scenario 16	200	50	50%	0.5 - 1	Discrete
Scenario 17	200	50	50%	0.5 - 1	Continuous and Discrete
Scenario 18	5000	100	50%	0.5 - 1	Continuous
Scenario 19	500	300	50%	0.5 - 1	Continuous

4.3 Output analysis

After running the code for one hundred iterations for each method, R generates an Excel spreadsheet containing columns for the iteration number, the selection method, the variables selected by each method along with their corresponding coefficients for each iteration. Additionally, it includes the time spent on the entire process and the mean time for each iteration.

To evaluate and compare the methods (Backward, Stepwise, lasso, and ssgl) in each scenario, several metrics were considered. These metrics include accuracy, sensitivity, specificity, and F1 score, along with the time spent to run. These metrics are commonly used in classification tasks to evaluate the performance of machine learning models. Depending on the specific objectives of the task and the class distribution in the dataset, one metric may be more relevant than others. Each of them will be discussed subsequently. For more details, see Hastie, Tibshirani, and Friedman (2008) and Witten, Frank, and Hall (2011).

- **Accuracy:**

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

where TP is the number of true positives, TN is the number of true negatives, FP is the number of false positives, and FN is the number of false negatives.

- Measures the overall correctness of a classification model;
- Represents the ratio of correctly predicted observations to the total observations;
- Provides a general understanding of how well a model performs across all classes. Commonly used when the classes in the dataset are balanced.

- **Sensitivity (True Positive Rate):**

$$\text{Specificity} = \frac{TN}{TN + FP}$$

- Measures the proportion of actual positive cases that were correctly identified by the model;
- Useful when the focus is on minimizing false negatives;
- Important in situations where missing a positive case can have serious consequences.

- **Specificity (True Negative Rate):**

$$\text{Specificity} = \frac{TN}{TN + FP}$$

- Measures the proportion of actual negative cases that were correctly identified by the model;
- Important when the emphasis is on minimizing false positives;
- Particularly useful in situations where false alarms are costly.

- **F1 Score:**

$$\text{F1 Score} = \frac{2 \cdot \text{Precision} \cdot \text{Sensitivity}}{\text{Precision} + \text{Sensitivity}}$$

- The harmonic mean of precision and recall;
- Provides a balance between precision and recall, considering both false positives and false negatives;
- Useful when there is an imbalanced class distribution or when both false positives and false negatives are equally important.

It is important to clarify that, in the context of this study, the indicators will measure each method's ability to select all variables where beta is greater than zero. For example, the method with the highest sensitivity will be the one that identifies the largest number of significant variables according to the parameters previously defined in the scenario.

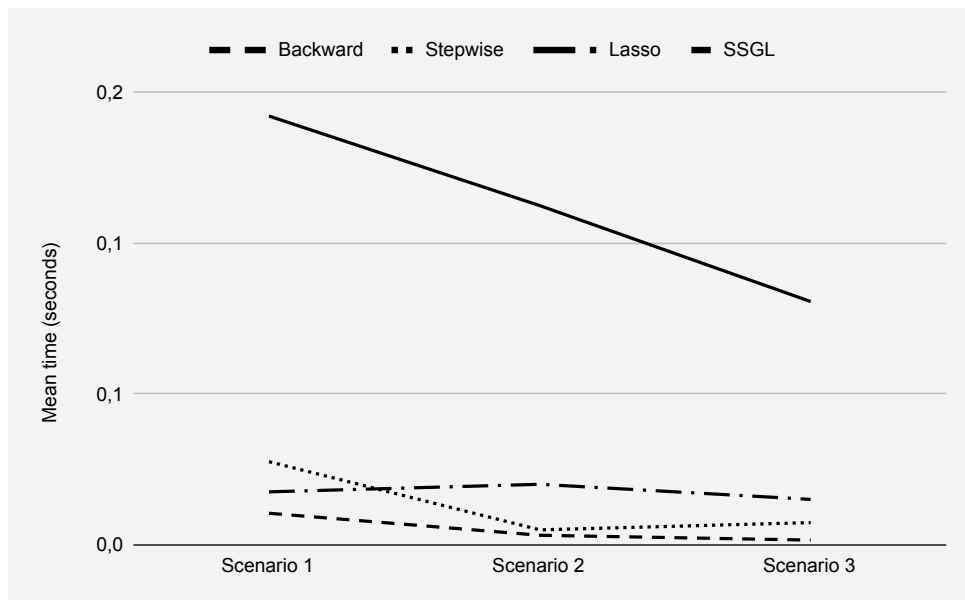
Table 2 – Parameter Settings for Scenarios 1, 2, and 3

Scenario	n	p	% variables $\beta > 0$	Beta range	Type of var
Scenario 1	20	10	70%	0 - 0.2	Continuous
Scenario 2	20	10	70%	0.5 - 1	Continuous
Scenario 3	20	10	70%	1 - 2	Continuous

4.3.1 Case 1: Beta Variation for Small Values of n and p

In Case 1, the scenarios involved were 1, 2, and 3, where n and p are considered to have small values (20 and 10, respectively) as shown in Table 2. The main objective was to evaluate the performance of the selection methods when n and p are small across different ranges of β values. The other indicators, such as the percentage of variables where β is greater than zero and the type of variables, as detailed in Table 2, remained consistent across these scenarios.

The total time to run the code in scenarios 1, 2, and 3 was 38, 36, and 32 seconds, respectively. The Figure 1 illustrates the mean time (in seconds) to run each iteration for each method and each scenario. Among the methods, Backward was the fastest, while SSGL was the slowest across all scenarios, though it's worth noting that even the slowest time was only about fifteen seconds. It is notable that the total time decreases as the β value increases from scenario 1 to scenario 3. This reduction in total time is primarily due to the decreased time required to run the SSGL method, as shown in Figure 1.

Figure 1 – Mean time per scenario for small values of n and p

Source: By the author

After analyzing the time to run, the methods will now be compared based on the metrics. To clarify and facilitate the analysis process, Table 3 was created. This table lists the methods in the rows for each scenario and the metrics in the columns. Additionally, some cells are highlighted in light green or light red to indicate the best-performing method or the worst-performing method

for each metric in each scenario. If the values are too similar for a given metric, the cells are not colored. For example, Lasso performed better than the other methods in the Sensitivity metric in Scenario 1, so its cell is highlighted in light green. However, in Scenarios 2 and 3, the Sensitivity metric is too similar for all methods, so there are no cells highlighted.

Backward and Stepwise performed similarly in the three scenarios, obtaining nearly identical results and improving in Accuracy, Sensitivity, and F1 score from the 1st to the 3rd scenario. In other words, these methods present better results when β values are larger. Lasso, in turn, significantly improved Sensitivity from the 1st to the 3rd scenario, although Specificity dropped at the same time. SSGL, on the other hand, showed clear improvement in all metrics from the 1st to the 3rd scenario (Table 3).

Table 3 – Performance Metrics for Scenarios 1, 2, and 3

Scenario	Method	Accuracy	Sensitivity	Specificity	F1 Score
Scenario 1	Backward	52.9%	47.0%	66.7%	58.3%
	Stepwise	52.9%	47.1%	66.3%	58.4%
	Lasso	42.8%	27.4%	78.7%	40.2%
	SSGL	50.6%	47.6%	57.7%	57.4%
Scenario 2	Backward	89.2%	99.1%	66.0%	92.8%
	Stepwise	89.2%	99.1%	66.0%	92.8%
	Lasso	75.6%	99.9%	19.0%	85.1%
	SSGL	80.2%	92.6%	51.3%	86.7%
Scenario 3	Backward	89.9%	100.0%	66.3%	93.3%
	Stepwise	89.9%	100.0%	66.3%	93.3%
	Lasso	74.9%	100.0%	16.3%	84.8%
	SSGL	95.8%	97.3%	92.3%	97.0%

4.3.2 Case 2: Beta Variation for Large Values of n and p

Case 2 includes scenarios 4, 5, and 6 as shown in Table 4. This table indicates that, similar to Case 1, the only varying metric was the β values. The key difference in this case is that n and p have larger values compared to Case 1 (500 and 100, respectively).

Table 4 – Parameter Settings for Scenarios 4, 5, and 6

Scenario	n	p	% variables $\beta > 0$	Beta range	Type of var
Scenario 4	500	100	50%	0 - 0.2	Continuous
Scenario 5	500	100	50%	0.5 - 1	Continuous
Scenario 6	500	100	50%	1 - 2	Continuous

The total time to run the code in scenarios 4, 5, and 6 was 79, 51, and 47 minutes, respectively. Figure 2 illustrates the mean time (in seconds) to run each iteration for each method and each scenario. Among the methods, Lasso was the fastest, with times approaching zero in all

scenarios. SSGL, in turn, shows an interesting trend, with a significant decrease in time spent (nearly 65%) from the first scenario to the second, and a further reduction in the third scenario. As in Case 1, it is notable that the total time decreases as the β value increases from the first scenario to the third, particularly due to the performance of the SSGL method (Figure 2).

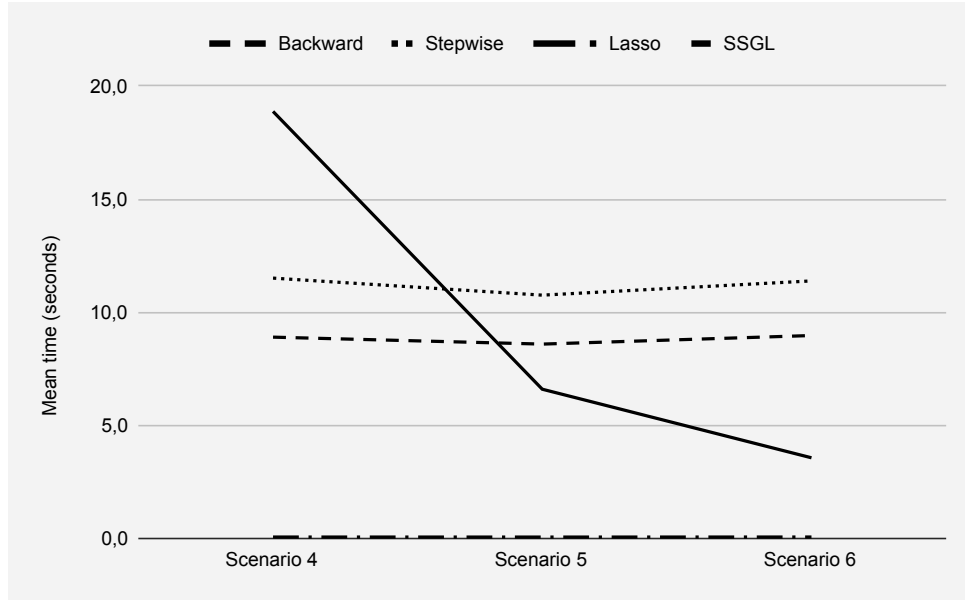


Figure 2 – Mean Time per Scenario across Varying Beta Strengths

Source: By the author

After analyzing the running times of the methods, attention now turns to their performance metrics based on Table 5. The Backward and Stepwise methods performed similarly across all three scenarios, as they did in Case 1, obtaining nearly identical results. From the 5th to the 6th scenario, there was no improvement in their performance as β values increased. In scenario 5, both methods can be considered the best in all metrics.

Lasso exhibited strong sensitivity performance across all scenarios. SSGL, on the other hand, displayed a notable improvement in the 6th scenario with the largest β value, similar to its performance in Case 1. In this scenario, SSGL emerged as the top-performing method across all metrics. Furthermore, it's worth mentioning that all methods achieved one hundred percent sensitivity in scenarios 5 and 6, indicating that they successfully identified all important variables as defined previously (Table 5)

4.3.3 Case 3: Beta Variation for Increasing n value

Scenarios 10, 11, and 12 were created to analyze Case 3. In this case, compared to Case 2, the p value remained the same, while n was doubled. To clarify, Case 2 serves as the baseline, with Cases 3 and 4 designed to analyze the behavior of β variation by changing the parameters p and n . Case 3 examines the effect of increasing n , whereas Case 4 explores the effect of increasing p compared to Case 2. These scenarios are demonstrated in Table 6.

Table 5 – Performance Metrics for Scenarios 1, 2, and 3

Scenario	Method	Accuracy	Sensitivity	Specificity	F1 Score
Scenario 4	Backward	82.5%	85.8%	79.2%	83.1%
	Stepwise	82.4%	85.9%	79.0%	83.0%
	Lasso	68.7%	94.6%	42.3%	75.3%
	SSGL	71.4%	90.7%	51.8%	76.2%
Scenario 5	Backward	89.4%	100.0%	78.9%	90.4%
	Stepwise	89.4%	100.0%	78.9%	90.4%
	Lasso	67.9%	100.0%	35.1%	75.9%
	SSGL	61.5%	100.0%	23.7%	72.0%
Scenario 6	Backward	89.4%	100.0%	78.9%	90.4%
	Stepwise	89.4%	100.0%	78.9%	90.4%
	Lasso	67.9%	100.0%	35.1%	75.9%
	SSGL	96.4%	100.0%	92.9%	96.5%

Table 6 – Parameter Settings for Scenarios 10, 11, and 12

Scenario	n	p	% variables $\beta > 0$	Beta range	Type of var
Scenario 10	1000	100	50%	0 - 0.2	Continuous
Scenario 11	1000	100	50%	0.5 - 1	Continuous
Scenario 12	1000	100	50%	1 - 2	Continuous

Figure 3 illustrates the mean time (in seconds) to run each iteration for each method and each scenario. The total time to run the code in scenarios 10, 11, and 12 was approximately 2, 1.8, and 1.5 hours, respectively, showing a similar decreasing trend as observed in previous cases.

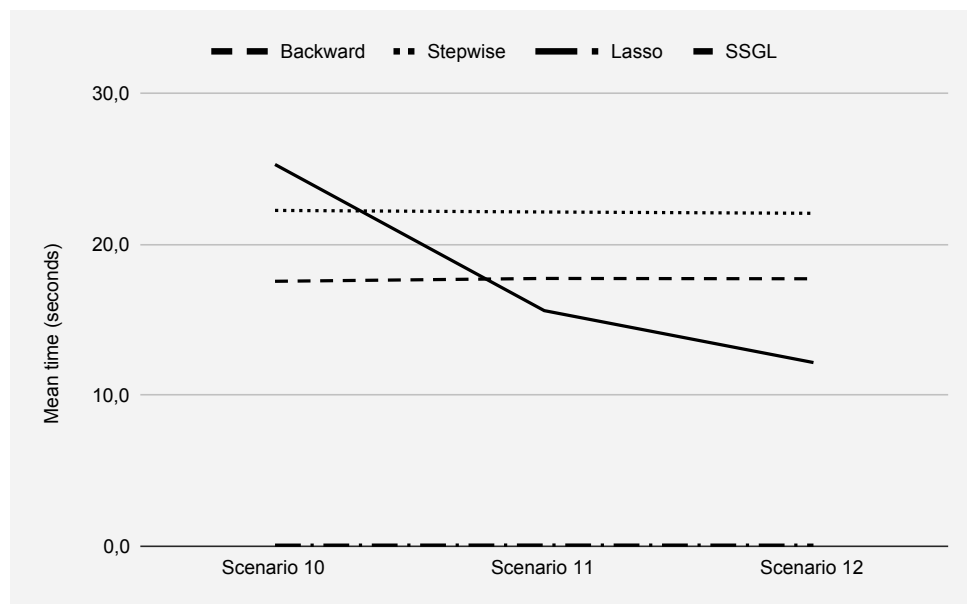
As shown in Figure 3, Lasso continued to be the fastest method to run in each iteration. The SSGL, similar to Case 2, required the most time to run when the β value was small but showed improved performance in the subsequent scenarios, spending less time than Backward and Stepwise methods.

Examining the performance metrics, the Backward method demonstrated slightly better results compared to Stepwise, and overall, it outperformed the other methods. Lasso surpassed Backward only in terms of sensitivity in scenario 10, as shown in Table 7.

SSGL showed a decline in its performance metrics—accuracy, specificity, and F1 score—from scenario 10 to scenario 12 as the β value increased. This pattern was not observed in the previous cases. Conversely, Lasso exhibited an improvement in all metrics across these scenarios, except in specificity (Table 7).

4.3.4 Case 4: Beta Variation for Increasing p value

In Case 4, scenarios 7, 8, and 9 are involved and are presented in Table 8. The key difference compared to Case 2 is that the p value is doubled, while n remains the same.

Figure 3 – Mean Time per Scenario across Varying Beta Strengths and Large n Values

Source: By the author

Table 7 – Performance Metrics for Scenarios 10, 11, and 12

Scenario	Method	Accuracy	Sensitivity	Specificity	F1 Score
Scenario 10	Backward	86.0%	91.2%	80.7%	86.7%
	Stepwise	86.0%	91.3%	80.7%	86.7%
	Lasso	68.2%	97.0%	39.4%	75.3%
	SSGL	73.4%	94.6%	52.1%	78.0%
Scenario 11	Backward	91.0%	100.0%	81.9%	91.7%
	Stepwise	90.9%	100.0%	81.8%	91.7%
	Lasso	68.3%	100.0%	36.5%	75.9%
	SSGL	70.6%	100.0%	41.3%	77.3%
Scenario 12	Backward	91.0%	100.0%	81.9%	91.7%
	Stepwise	90.9%	100.0%	81.8%	91.7%
	Lasso	70.6%	100.0%	41.3%	77.3%
	SSGL	60.8%	100.0%	21.5%	71.8%

Table 8 – Parameter Settings for Scenarios 7, 8, and 9

Scenario	n	p	% variables $\beta > 0$	Beta range	Type of var
Scenario 7	500	200	50%	0 - 0.2	Continuous
Scenario 8	500	200	50%	0.5 - 1	Continuous
Scenario 9	500	200	50%	1 - 2	Continuous

The total time to run the code in scenarios 7, 8, and 9 was approximately 11, 8, and 7.5 hours, respectively, showing a similar decreasing trend as observed in previous cases. Figure 4 illustrates the mean time (in seconds) to run each iteration for each method and each scenario.

Notably, based on Figure 4, Lasso was the fastest method to run in each iteration, followed by SSGL and Backward. Stepwise was the slowest of all. It is important to mention that all methods showed a tendency to reduce their running time as the β value increased, which is especially evident for the SSGL method when examining Figure 4. Additionally, it is relevant to note that traditional methods like Backward and Stepwise required more time to run in all scenarios as the p value increased.

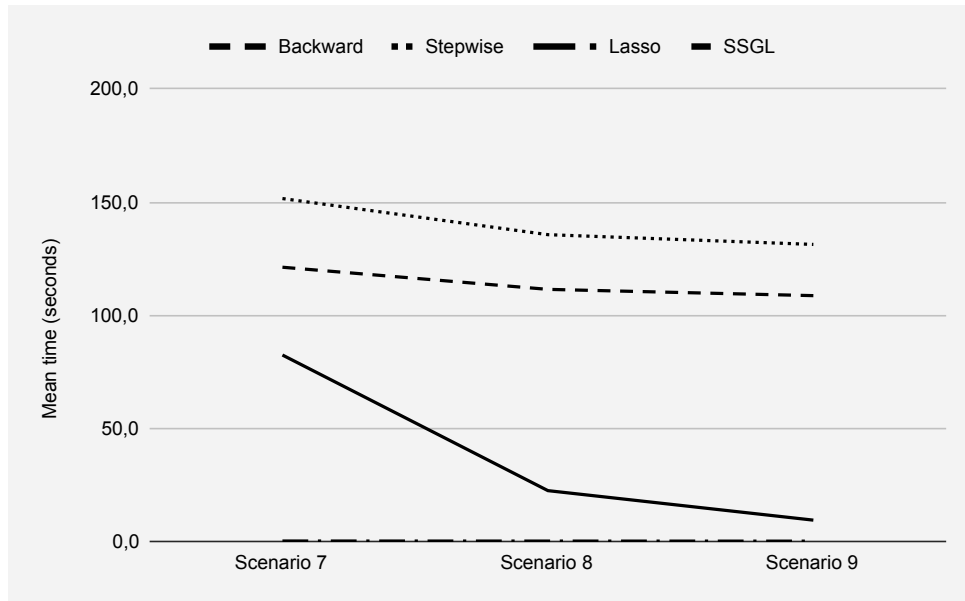


Figure 4 – Mean Time per Scenario across Varying Beta Strengths and Large p Values

Source: By the author

Regarding the performance metrics, Backward and Stepwise methods continued to show similar behavior, even though Stepwise took 21% more time to run than Backward. Both methods demonstrated the best performance in scenarios 7 and 8 for accuracy, specificity, and F1 score, as shown in Table 9. Sensitivity achieved the highest level for all methods.

On the other hand, Lasso and SSGL were the best-performing methods in scenario 9, where the β value is larger. SSGL, in particular, stood out in this scenario, achieving the highest values across all metrics. However, SSGL exhibited a drop in its specificity value from scenario 7 to scenario 8, achieving the smallest value in all scenarios. This was followed by an increase in this metric in scenario 9, mirroring its performance pattern observed in Case 2 (Table 9).

4.3.5 Case 5: Variation of the Percentage of Significant Variables

Scenarios 13, 14, and 15 were created to analyze case 5, as presented in Table 10. In this case, the main objective was to observe the behavior of the methods when the percentage of

Table 9 – Performance Metrics for Scenarios 7, 8, and 9

Scenario	Method	Accuracy	Sensitivity	Specificity	F1 Score
Scenario 7	Backward	79.2%	85.7%	72.7%	80.4%
	Stepwise	79.1%	85.8%	72.5%	80.4%
	Lasso	68.0%	93.5%	42.4%	74.5%
	SSGL	70.0%	92.2%	47.8%	75.4%
Scenario 8	Backward	86.8%	100.0%	73.6%	88.3%
	Stepwise	86.7%	100.0%	73.4%	88.2%
	Lasso	66.3%	100.0%	32.6%	74.8%
	SSGL	54.2%	100.0%	8.4%	68.6%
Scenario 9	Backward	86.0%	100.0%	72.0%	87.7%
	Stepwise	85.9%	100.0%	71.9%	87.7%
	Lasso	87.9%	100.0%	75.9%	89.2%
	SSGL	90.6%	100.0%	81.2%	91.4%

significant variables changes while all other parameters remain fixed. It is important to note that the other parameters were chosen with the aim of optimizing the runtime and avoiding extreme values. For example, n and p were selected to be neither too small nor too large, balancing the computational time. Similarly, the values of β were chosen to be moderate, avoiding extreme values (Table 10).

Table 10 – Parameter Settings for Scenarios 13, 14, and 15

Scenario	n	p	% variables $\beta > 0$	Beta range	Type of var
Scenario 13	200	50	20%	0.5 - 1	Continuous
Scenario 14	200	50	50%	0.5 - 1	Continuous
Scenario 15	200	50	70%	0.5 - 1	Continuous

Examining the total time spent to run the scenarios, Scenarios 13, 14, and 15 took 7.8, 5.4, and 4.3 minutes, respectively. This indicates a decrease in time as the percentage of significant variables increased. It's important to note that β remained fixed across all scenarios, unlike in the others cases where it varied. Figure 5 illustrates the mean time (in seconds) to run each iteration for each method and each scenario.

The lasso method consistently stood out, taking less than a second to run in all scenarios. The other methods showed a decrease in the mean time per iteration as the percentage of significant variables increased, with the SSGL method displaying a particularly notable decrease, as seen in Figure 5.

As can be observed in Table 11, all scenarios showed a sensitivity metric of one hundred percent, indicating a strong capacity to identify the relevant variables. One interesting aspect is that the SSGL method went from being the best performer to the worst in the specificity metric as the percentage of significant variables increased from the 13th to the 15th scenario. Conversely, the lasso method did not show significant variation in its metrics across the scenarios. The

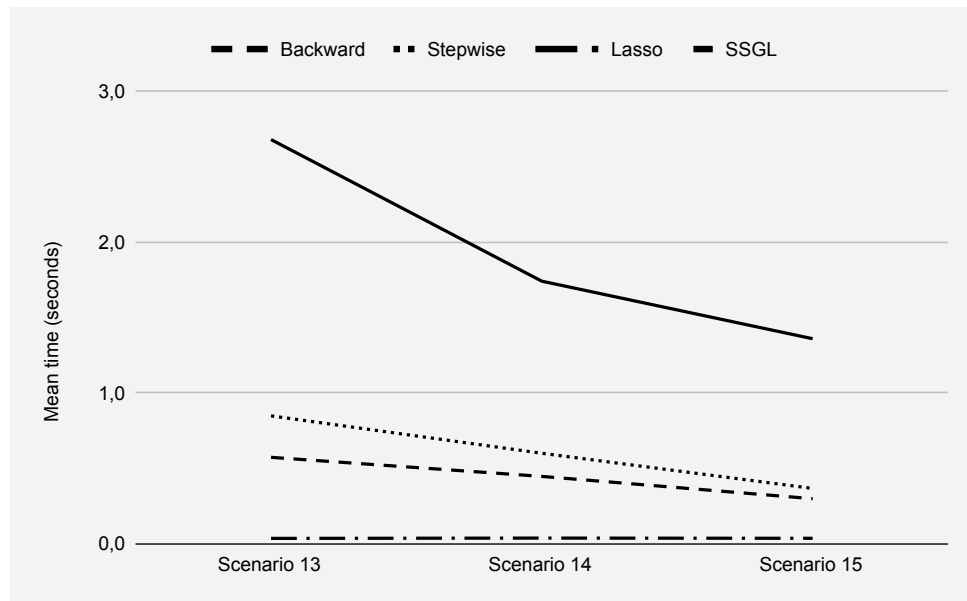


Figure 5 – Mean Time per Scenario: Percentage of Significant Variables

Source: By the author

Backward and Stepwise methods performed similarly, with some improvement in accuracy and F1 score from the 13th to the 15th scenario, particularly standing out in Scenario 15 compared to the other methods (Table 11).

Table 11 – Performance Metrics for Scenarios 13, 14, and 15

Scenario	Method	Accuracy	Sensitivity	Specificity	F1 Score
Scenario 13	Backward	82.5%	100.0%	78.1%	69.5%
	Stepwise	82.3%	100.0%	77.8%	69.3%
	Lasso	69.4%	100.0%	61.7%	56.6%
	SSGL	90.4%	100.0%	88.0%	80.6%
Scenario 14	Backward	88.9%	100.0%	77.9%	90.0%
	Stepwise	88.9%	100.0%	77.8%	90.0%
	Lasso	67.4%	100.0%	34.9%	75.4%
	SSGL	66.0%	100.0%	32.0%	74.6%
Scenario 15	Backward	93.2%	100.0%	77.2%	95.3%
	Stepwise	93.2%	100.0%	77.2%	95.3%
	Lasso	76.7%	100.0%	22.2%	85.7%
	SSGL	75.7%	100.0%	18.9%	85.2%

4.3.6 Case 6: Variation in the Type of Variables

Case 6 contemplates scenarios 14, 16, and 17 with all parameters fixed except for the type of variables: continuous, discrete, or mixed (half discrete, half continuous), as presented in Table 12.

Table 12 – Parameter Settings for Scenarios 14, 16, and 17

Scenario	n	p	% variables $\beta > 0$	Beta range	Type of var
Scenario 14	200	50	50%	0.5 - 1	Continuous
Scenario 16	200	50	50%	0.5 - 1	Discrete
Scenario 17	200	50	50%	0.5 - 1	Continuous and Discrete

The total time to run scenarios 14, 16, and 17 was 5.4, 6.7, and 5.7 minutes, respectively. This indicates an increase in time followed by a decrease. It is evident that the time is impacted when the variables involved are solely discrete.

Figure 6 illustrates the mean time (in seconds) to run each iteration for each method and each scenario. The increase in time in scenario 16, where the variables are all discrete, is particularly due to the SSGL method, which seems to be more affected by discrete variables. The lasso method remained the fastest, followed by Backward and Stepwise.

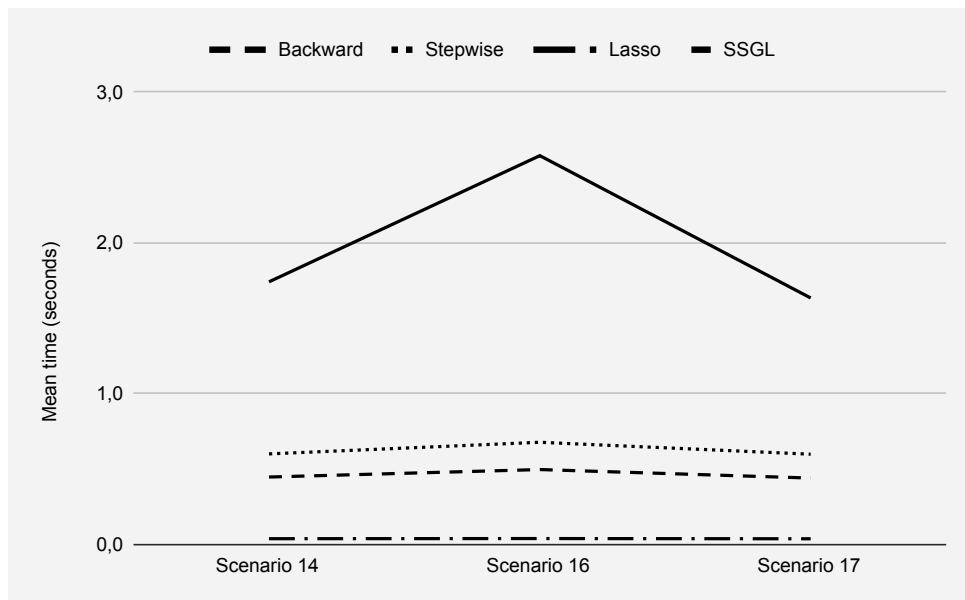


Figure 6 – Mean Time per Scenario: Variation Across Types of Variables

Source: By the author

Based on Table 13, all metrics across all scenarios showed a high level of sensitivity, regardless of the type of variables. Both Backward and Stepwise methods exhibited better performance than the others across all metrics and scenarios. It's important to note that this performance may be influenced more by the values of p and n rather than the type of variables.

The key observation here is that there isn't a significant variation in the metrics among the scenarios, except for SSGL, which shows an improvement in performance metrics in scenario 16 (discrete variables only) compared to the others (Table 13).

Table 13 – Performance Metrics for Scenarios 14, 16, and 17

Scenario	Method	Accuracy	Sensitivity	Specificity	F1 Score
Scenario 14	Backward	88.9%	100.0%	77.9%	90.0%
	Stepwise	88.9%	100.0%	77.8%	90.0%
	Lasso	67.4%	100.0%	34.9%	75.4%
	SSGL	66.0%	100.0%	32.0%	74.6%
Scenario 16	Backward	89.3%	100.0%	78.5%	90.3%
	Stepwise	89.2%	100.0%	78.4%	90.3%
	Lasso	69.0%	100.0%	38.0%	76.3%
	SSGL	77.4%	100.0%	54.8%	81.6%
Scenario 17	Backward	89.4%	100.0%	78.8%	90.4%
	Stepwise	89.4%	100.0%	78.8%	90.4%
	Lasso	68.1%	100.0%	36.2%	75.8%
	SSGL	68.0%	100.0%	36.0%	75.8%

4.3.7 Case 7: Increase in n Value

This case was designed to analyze and compare scenarios 5, 11, and 18, focusing on variations in n while keeping other parameters fixed. Table 14 illustrates these variations, highlighting scenario 18 with a significantly higher n value compared to the others. This setup was specifically designed to observe behaviors in more extreme scenarios.

Table 14 – Parameter Settings for Scenarios 5, 11, and 18

Scenario	n	p	% variables $\beta > 0$	Beta range	Type of var
Scenario 5	500	100	50%	0.5 - 1	Continuous
Scenario 11	1000	100	50%	0.5 - 1	Continuous
Scenario 18	5000	100	50%	0.5 - 1	Continuous

The time spent to run the scenarios shows an increase from scenario 5 to 11, and becomes significantly longer, nearly five times more, from 11 to 18. Specifically, it takes 0.8, 1.8, and 9.1 hours, respectively. Thus, an increase in n results in a corresponding increase in time.

Figure 7 shows the mean time (in seconds) to run each iteration for each method in each scenario. The lasso method remains unaffected, taking less than one second to run in all cases. SSGL, Backward, and Stepwise share similar behavior in the first two scenarios, with lasso being the fastest, followed by SSGL, and then Backward. However, in the 18th scenario, Backward runs faster than SSGL, as clearly shown in Figure 7.

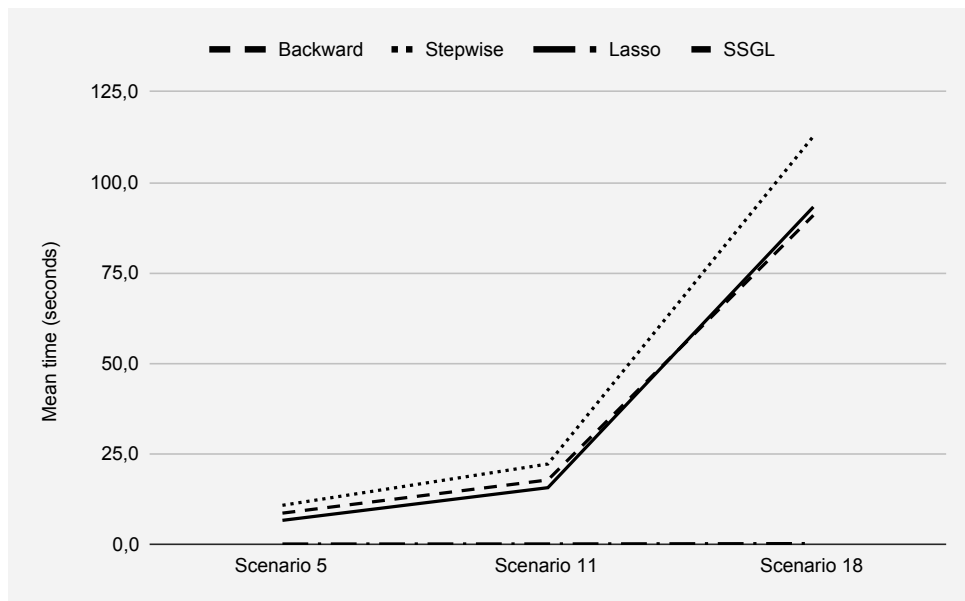


Figure 7 – Mean time per scenario: Increase in n Value

Source: By the author

In this case, sensitivity consistently shows good performance for all metrics across all methods. Regarding the other metrics, they notably improved or at least remained the same from

scenario 5 to 18 for all methods. In other words, with the increase in n , the methods can make better decisions. However, Backward and Stepwise methods demonstrated superior performance compared to the other two methods in all scenarios, as seen in Table 15.

Table 15 – Performance Metrics for Scenarios 5, 11, and 18

Scenario	Method	Accuracy	Sensitivity	Specificity	F1 Score
Scenario 5	Backward	89.4%	100.0%	78.9%	90.4%
	Stepwise	89.4%	100.0%	78.9%	90.4%
	Lasso	67.9%	100.0%	35.1%	75.9%
	SSGL	61.5%	100.0%	23.7%	72.0%
Scenario 11	Backward	91.0%	100.0%	81.9%	91.7%
	Stepwise	90.9%	100.0%	81.8%	91.7%
	Lasso	68.3%	100.0%	36.5%	75.9%
	SSGL	70.6%	100.0%	41.3%	77.3%
Scenario 18	Backward	91.8%	100.0%	83.6%	92.4%
	Stepwise	91.8%	100.0%	83.6%	92.4%
	Lasso	72.0%	100.0%	44.0%	78.1%
	SSGL	75.1%	100.0%	50.2%	80.1%

4.3.8 Case 8: Increase in p Value

Unlike case 7, this one includes scenarios 5, 8, and 19, which vary p while keeping other parameters fixed. Table 16 illustrates these variations, showing that p increased twofold in scenario 8 and threefold in scenario 19 compared to scenario 5. This setup was also specifically designed to observe behaviors in more extreme scenarios.

Table 16 – Parameter Settings for Scenarios 5, 8, and 19

Scenario	n	p	% variables $\beta > 0$	Beta range	Type of var
Scenario 5	500	100	50%	0.5 - 1	Continuous
Scenario 8	500	200	50%	0.5 - 1	Continuous
Scenario 19	500	300	50%	0.5 - 1	Continuous

The time spent to run the scenarios 5, 8, and 19 was 0.8, 8.1, and 31.9 hours, respectively. It is noteworthy that the time increased fortyfold from scenario 5 to scenario 19, demonstrating that the increase in p significantly impacts the run time, even more so than the increase in n .

Figure 8 clearly shows that increasing the number of candidate variables (p) directly affects the run time for the Stepwise and Backward methods. SSGL is also affected, though to a lesser extent. It's important to mention that the impact on run time for SSGL is much more significant with variations in n than in p .

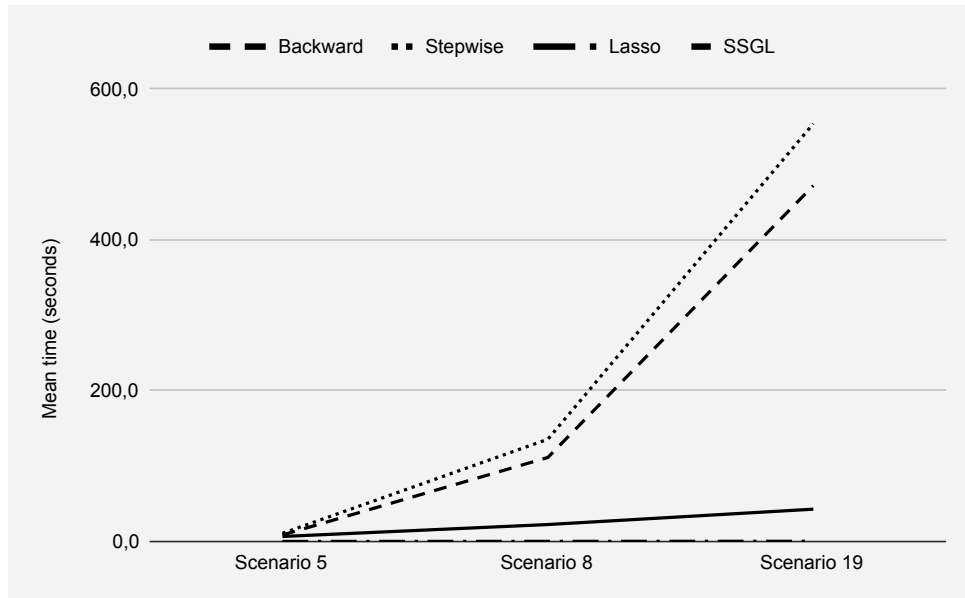


Figure 8 – Mean time per scenario: Increase in p Value

Source: By the author

Once again, sensitivity demonstrated good performance across all scenarios and methods, as presented in Table 17. In the first two scenarios, Backward and Stepwise stood out from the

other methods, showing the best performance. However, in the 19th scenario, SSGL emerged as the best method.

It is important to note that with the increment in p , Backward, Stepwise, and lasso exhibited a decrease in their performance. SSGL, on the other hand, showed a significant drop in performance from scenario 5 to 8, achieving only 8.4% in the specificity metric, but rebounded to a high level of performance in the 19th scenario. This behavior is quite interesting and indicates that this method can be very volatile, with some metrics being highly sensitive to changes in the number of variables (Table 17).

Table 17 – Performance Metrics for Scenarios 5, 8, and 19

Scenario	Method	Accuracy	Sensitivity	Specificity	F1 Score
Scenario 5	Backward	89.4%	100.0%	78.9%	90.4%
	Stepwise	89.4%	100.0%	78.9%	90.4%
	Lasso	67.9%	100.0%	35.1%	75.9%
	SSGL	61.5%	100.0%	23.7%	72.0%
Scenario 8	Backward	86.8%	100.0%	73.6%	88.3%
	Stepwise	86.7%	100.0%	73.4%	88.2%
	Lasso	66.3%	100.0%	32.6%	74.8%
	SSGL	54.2%	100.0%	8.4%	68.6%
Scenario 19	Backward	81.9%	100.0%	63.7%	84.6%
	Stepwise	81.5%	100.0%	63.1%	84.4%
	Lasso	64.2%	100.0%	28.5%	73.7%
	SSGL	94.8%	100.0%	89.6%	95.1%

4.4 Real example: Anthropometric Data

After simulating the scenarios, a real data example was applied to test the selection methods and analyze their behavior, ultimately aiding in determining which method to choose.

In this example, will be conduct an analysis of anthropometric data and explore the relationships between the variables. In other words, we will examine how weight interacts with other variables such as height, neck circumference, chest circumference, abdominal circumference, hip circumference, among others. The dataset contains 252 observations of 18 variables, closely fitting scenarios 13, 14, and 15 in terms of n and p values from the simulation.

This dataset was used in the Applied Statistics specialization course to illustrate multicollinearity but was not employed for modeling or selection methods, as will be done here. Additionally, all initial analyses in this example were conducted using Python to explore additional functions and alternative graphical visualization methods not covered in the specialization course. Since Python lacks specific selection method functions, this aspect will continue to be addressed using R.

4.4.1 Initial analysis

At first, the database was saved as `data2` using the Python function `read_csv()` from the pandas library. Then, the Python function `head()` was used to display a preview of the database, and `columns` was used to show all the variables (Listing 4.2, line 6).

Listing 4.2 – Data structure

```

1 Python input:
2 import pandas as pd
3 data2 = pd.read_csv("data_gordura.csv", header=0, decimal=".", sep=";")
4 print(data2.head())
5 col_names = data2.columns
6 print(col_names)
7
8 Python output:
9 Index(['brozek', 'siri', 'density', 'age', 'weight', 'height', 'adipos_IMC',
10        'free', 'neck', 'chest', 'abdom', 'hip', 'thigh', 'knee', 'ankle',
11        'biceps', 'forearm', 'wrist'],
12       dtype='object')
```

After examining the data structure and the available variables in the database, a correlation plot (Figure 9) was generated to gain insights into the relationships between them. It is noticeable that some variables exhibit weak correlations with others, as indicated by values close to zero.

For example, age shows a weak correlation with almost all variables, close to zero. Conversely, density exhibits a negative correlation with all variables, close to -1. On the other hand, weight exhibits one of the strongest positive correlations, close to 1, with a great part of the variables such as neck, chest, abdom, hip, thigh, and knee.

Besides that, the code used to plot is in Listing 4.3, and the functions were imported from the libraries *seaborn*, *matplotlib.pyplot*, and *numpy*. Following the steps, the variables were joined in a data frame. After that, a correlation matrix was constructed using the function *corr()*, and some other adjustments were made to get the right dimensions for the plot.

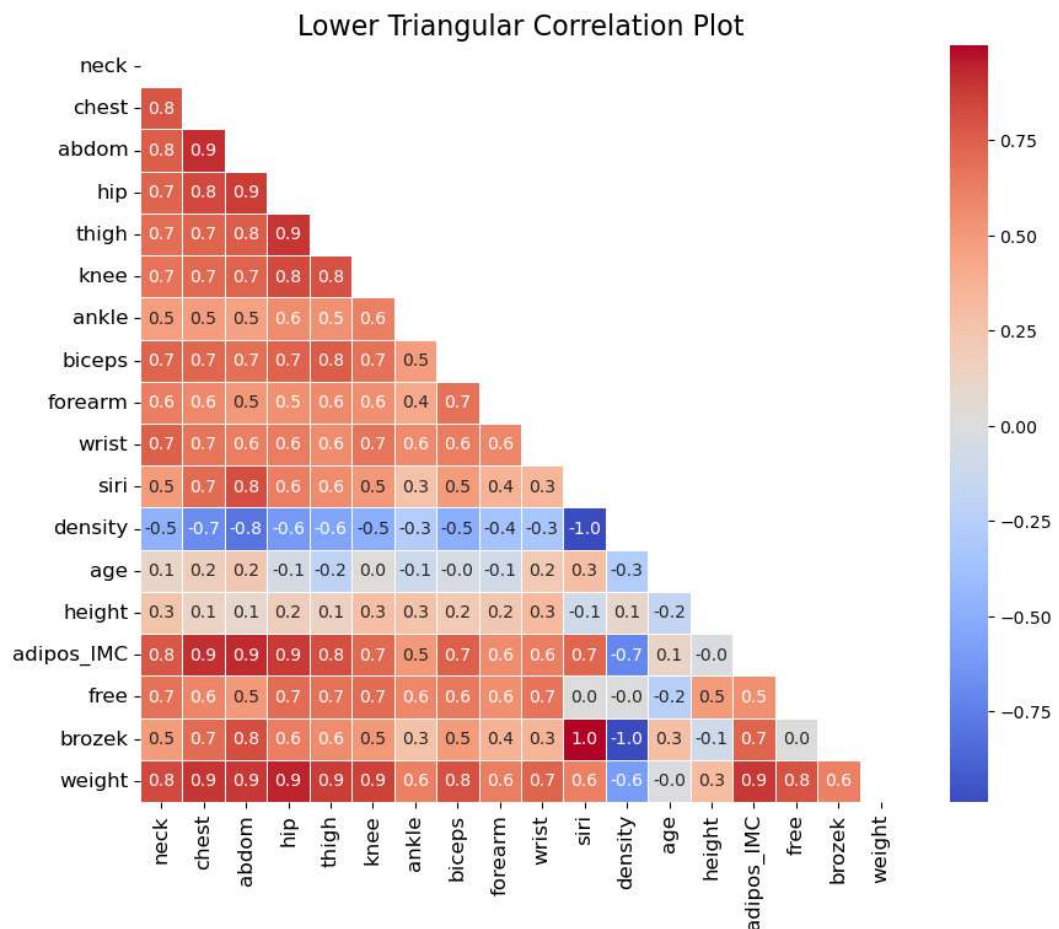


Figure 9 – Correlation matrix - anthropometric data

Source: By the author

Listing 4.3 – Correlation plot

```

1 Python input:
2 import seaborn as sns
3 import matplotlib.pyplot as plt
4 import numpy as np
5
6 variable = ["neck", "chest", "abdom", "hip", "thigh", "knee",
7            "ankle", "biceps", "forearm", "wrist", "siri", "density",
8            "age", "height", "adipos_IMC", "free", "brozek", "weight"]

```

```

9
10 subset_data = data2[variable]
11
12 correlation_matrix = subset_data.corr()
13
14 mask = np.triu(np.ones_like(correlation_matrix, dtype=bool))
15 plt.figure(figsize=(10, 8))
16 sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".1f",
17             linewidths=0.5, mask=mask)
18 plt.title('Lower Triangular Correlation Plot', fontsize=16)
19 plt.xticks(fontsize=12)
20 plt.yticks(fontsize=12)
21 plt.show()

```

So, after analyzing the variables, it's time to fit a model. Firstly, the *statsmodels.api* library was imported, which provides numerous statistical functions in Python. Then, the *fit()* function was utilized using the OLS (Ordinary Least Squares) class. All the code is shown in Listing 4.4.

For clear information and decision-making, the model output has been separated into tables. Table 18 presents the results of the model fit, including R-squared, F-statistic, AIC, BIC, among other indicators. It is important to note that when R-squared and Adj. R-squared have results equal to 1.0, it indicates "overfitting" of the model. This suggests that while the model fits the current data well, it may not generalize well to other data.

Table 19 presents columns for coefficients, standard error, t-value, p-value, and confidence interval for each covariate. Notably, certain variables are highlighted in bold based on their p-values to indicate their significance to the model, with p-values considered "small" (less than 0.05).

Listing 4.4 – Model1

```

1 Python input:
2 import statsmodels.api as sm
3 covariable = ["neck", "chest", "abdom", "hip", "thigh", "knee",
4             "ankle", "biceps", "forearm", "wrist", "siri", "density",
5             "age", "height", "adipos_IMC", "free", "brozek"]
6 data3 = data2[covariable]
7 data3 = sm.add_constant(data3)
8 X = data3[covariable]
9 Y = data2['weight']
10 modelo1 = sm.OLS(Y, X).fit()
11 print(modelo1.summary())

```

Table 18 – OLS Regression Results - Model1

Dep. Variable	weight
Model	OLS
Method	Least Squares
No. Observations	252
Df Residuals	235
Df Model	17
Covariance Type	nonrobust
R-squared (uncentered)	1.000
Adj. R-squared (uncentered)	1.000
F-statistic	5.163×10^4
Prob (F-statistic)	0.00
Log-Likelihood	-631.58
AIC	1297.
BIC	1357.

Table 19 – Regression coefficients - Model1

Variable	Coefficient	Standard Error	t	p Value	[95% CI]
neck	0.2378	0.169	1.407	0.161	(-0.095, 0.571)
chest	0.0120	0.077	0.154	0.877	(-0.141, 0.165)
abdom	0.0754	0.079	0.956	0.340	(-0.080, 0.231)
hip	0.4307	0.102	4.208	0.000	(0.229, 0.632)
thigh	-0.2674	0.107	-2.493	0.013	(-0.479, -0.056)
knee	0.2671	0.178	1.497	0.136	(-0.084, 0.619)
ankle	0.0122	0.160	0.076	0.939	(-0.303, 0.328)
biceps	0.0256	0.125	0.205	0.837	(-0.220, 0.272)
forearm	-0.2987	0.145	-2.055	0.041	(-0.585, -0.012)
wrist	-0.2117	0.394	-0.538	0.591	(-0.987, 0.564)
siri	-0.7300	1.062	-0.687	0.493	(-2.823, 1.363)
density	-72.6164	10.240	-7.091	0.000	(-92.791, -52.442)
age	-0.0344	0.023	-1.484	0.139	(-0.080, 0.011)
height	0.1912	0.079	2.433	0.016	(0.036, 0.346)
adipos_IMC	1.0490	0.214	4.891	0.000	(0.626, 1.472)
free	1.0073	0.047	21.641	0.000	(0.916, 1.099)
brozek	2.2642	1.147	1.974	0.050	(0.005, 4.524)

Before assessing the significance of individual variables or employing any variable selection methods, it's crucial to analyze multicollinearity and examine whether the variables are correlated with each other. For this purpose, the *VIF()* (Variance Inflation Factor) function was employed in R.

Since the primary focus of this work does not involve discussing multicollinearity, the process was not elaborated upon. However, the criteria utilized for this analysis involved the following steps:

- Calculating the model and the VIF;
- Identifying the variable with the highest VIF;
- Removing the variable with the highest VIF and updating the model;
- Recalculating the VIF for the remaining variables;
- Repeating steps 2-4 until the VIF value for all remaining variables is less than 5.

After this process, the following covariates remained: neck, chest, knee, ankle, biceps, forearm, wrist, age, density, and height. The model with the remaining variables is shown in Listing 4.5. In Table 20, key information about the model such as the F-statistic, R-squared, Adj R-squared, AIC, and BIC are presented. An R-squared value of 99.8% indicates that a large portion of the variability in the dependent variable can be explained by the model.

Besides that, Table 21 shows the compiled information about all variables individually. By examining the p-values and assuming a significance level of 5%, it is possible to conclude that the remaining variables are significant to the model based on this test. Now, it is time to determine if the selection methods would indicate that all variables are necessary, or if any could be dropped from the model.

Listing 4.5 – Model2

```

1 Python input:
2 covariable2 = ["neck", "chest", "knee", "ankle", "biceps", "forearm", "
    wrist",
3             "age", "density", "height"]
4
5 model2 = sm.OLS(data2['weight'], data2[covariable2]).fit()
6
7 print(model2.summary())
```

After conducting initial analyses and applying the Variance Inflation Factor (VIF) process, various variable selection methods will be applied to this database to determine if the conclusions reached through analysis alone remain consistent. The methods used will be the same as those in the simulation, with the addition of some other methods that will be described in detail throughout the chapter.

Table 20 – OLS Regression Results - Model2

Dep. Variable	weight
Model	OLS
Method	Least Squares
No. Observations	252
Df Residuals	242
Df Model	10
Covariance Type	nonrobust
R-squared (uncentered)	0.998
Adj. R-squared (uncentered)	0.998
F-statistic	1.442×10^4
Prob (F-statistic)	0.00
Log-Likelihood	-862.71
AIC	1745.
BIC	1781.

Table 21 – Regression Coefficients - Model2

Variable	Coefficient	Standard Error	t	p Value	[95% CI]
neck	1.9547	0.383	5.106	0.000	1.201 - 2.709
chest	1.2979	0.111	11.720	0.000	1.080 - 1.516
knee	3.0519	0.337	9.069	0.000	2.389 - 3.715
ankle	0.7859	0.382	2.058	0.041	0.034 - 1.538
biceps	0.9689	0.281	3.447	0.001	0.415 - 1.523
forearm	-0.7917	0.344	-2.303	0.022	-1.469 - -0.115
wrist	2.7126	0.934	2.904	0.004	0.873 - 4.552
age	-0.3420	0.044	-7.782	0.000	-0.429 - -0.255
density	-234.2475	10.107	-23.177	0.000	-254.157 - -214.338
height	0.6056	0.147	4.106	0.000	0.315 - 0.896

4.4.2 Backward Stepwise Selection

Backward selection method was applied to the covariates using the *step()* function in R. However, in this scenario, the selection criterion was determined by the parameter *k*. *k* serves as a multiplier of the degrees of freedom used as a penalty criterion in selection methods such as Backward, forward, and Stepwise. According to the R help documentation, *k* can take the following values:

- When $k=2$, the criterion used is the AIC (Akaike Information Criterion);
- When $k=\log(n)$, where *n* is the number of observations, the criterion used is the BIC (Bayesian Information Criterion) or SBC (Schwarz Information Criterion).

Both AIC and BIC are model selection criteria used to compare different statistical

models. AIC tends to select more complex models, while BIC attempts to penalize complexity more severely. While AIC has a fixed value of k , for BIC it can vary depending on the number of observations.

In the R code provided in Listing 4.6, Backward selection is performed using both the AIC (line 2) and BIC criterion (line 7). The *model2* contains the covariates that remained after the VIF selection, which was explained in the previous section.

The final R output for both criteria is shown in Tables 22 and 23. In Table 22, it is evident that all initial variables are significant to the model and remained, whereas in Table 23, the variables "forearm" and "wrist" were dropped because they were not considered significant to the model based on BIC criteria.

Listing 4.6 – Step-Backward

```

1 R input (AIC):
2 model_Backward1 <- step(model2, direction = 'Backward', k = 2)
3
4 R input (BIC):
5 n = nrow(data2)
6 n
7 model_Backward2 <- step(model2, direction = 'Backward', k = log(n))

```

Table 22 – Output Backward - AIC

Variable	Df	Sum of Sq	RSS	AIC
<i>None</i>		13185	1019.3	
forearm	1	225.1	13410	1021.5
ankle	1	274.3	13459	1022.5
wrist	1	281.5	13466	1022.6
density	1	453.5	13638	1025.8
biceps	1	743.3	13928	1031.1
height	1	915.0	14100	1034.2
neck	1	1475.3	14660	1044.0
age	1	2334.8	15520	1058.3
knee	1	5241.0	18426	1101.6
chest	1	8392.1	21577	1141.4

Table 23 – Output Backward - BIC

Variable	Df	Sum of Sq	RSS	AIC
<i>None</i>		13611	1055.0	
density	1	315.6	13926	1055.3
ankle	1	476.5	14087	1058.2
biceps	1	694.7	14306	1062.1
height	1	1071.4	14682	1068.6
age	1	1956.9	15568	1083.4
neck	1	2019.8	15631	1084.4
knee	1	5757.0	19368	1138.4
chest	1	8672.4	22283	1173.7

4.4.3 Forward Stepwise Selection

Same as was done with Backward, the forward method was also applied using the parameter `k` as an argument of the function and with both criteria, AIC and BIC. In Listing 4.7, it is possible to see the structure of the code for both cases in lines 1 and 5. In the same Listing, the R output is shown on lines 9 and 15.

The conclusion is the same as Backward. Considering AIC as a criterion, all variables are significant to the model, but BIC penalizes the complexity and suggests dropping "forearm" and "wrist," as shown in lines 22 and 23 (Listing 4.7).

Listing 4.7 – Step-forward

```

1 R input (AIC):
2 model0 <- lm(data2$weight ~ 1)
3 model_forward1 <- step(model0, scope = ~ data2$neck + data2$chest + data2$
  knee +      data2$ankle + data2$biceps + data2$forearm + data2$wrist +
  data2$density +      data2$age + data2$height, direction = 'forward',
  k = 2)
4
5 R input (BIC):
6 model0 <- lm(data2$weight ~ 1)
7 model_forward2 <- step(model0, scope = ~ data2$neck + data2$chest + data2$
  knee +      data2$ankle + data2$biceps + data2$forearm + data2$wrist +
  data2$density +      data2$age + data2$height, direction = 'forward',
  k = log(n))
8
9 R output (AIC):
10 Step:  AIC=1019.26
11 data2$weight ~ data2$chest + data2$knee + data2$neck + data2$age +
12   data2$height + data2$biceps + data2$ankle + data2$density +
13   data2$wrist + data2$forearm
14

```

```

15 R output (BIC)
16 Step:  AIC=1055.04
17 data2$weight ~ data2$chest + data2$knee + data2$neck + data2$age +
18     data2$height + data2$biceps + data2$ankle + data2$density
19
20           Df Sum of Sq  RSS   AIC
21 <none>                13611 1055.0
22 + data2$wrist      1    200.91 13410 1056.8
23 + data2$forearm    1    144.50 13466 1057.9

```

4.4.4 Stepwise Selection

In Listing 4.8, the Stepwise method is presented for this example using the same criteria as in forward and Backward. In lines 1 and 7, the R input and output for AIC are shown, respectively, while in lines 4 and 12, the same is demonstrated for BIC. Additionally, the final R results are displayed in Tables 24 and 25 for both cases.

Similar to the other methods, with AIC criteria in Stepwise, all variables are significant for the model. Conversely, BIC criteria excluded two variables, as evident in the same Listing and Tables. For instance, in line 14 of the Listing, "forearm" and "wrist" are not included, and in Table 25 (BIC criteria), the same variables are marked with "+" indicating their exclusion from the model.

Listing 4.8 – Stepwise

```

1 R input (AIC):
2 model_step <- step(model2, direction = 'both', k = 2)
3
4 R input (BIC):
5 model_step <- step(model2, direction = 'both', k = log(n))
6
7 R output (AIC):
8 Start:  AIC=1019.26
9 weight ~ neck + chest + knee + ankle + biceps + forearm + wrist +
10     age + density + height
11
12 R output (BIC):
13 Step:  AIC=1055.04
14 weight ~ neck + chest + knee + ankle + biceps + age + density +
15     height

```

Table 24 – Output Stepwise - AIC

Change	Df	Sum of Sq	RSS	AIC
<i>None</i>			13185	1019.3
- forearm	1	225.1	13410	1021.5
- ankle	1	274.3	13459	1022.5
- wrist	1	281.5	13466	1022.6
- density	1	453.5	13638	1025.8
- biceps	1	743.3	13928	1031.1
- height	1	915.0	14100	1034.2
- neck	1	1475.3	14660	1044.0
- age	1	2334.8	15520	1058.3
- knee	1	5241.0	18426	1101.6
- chest	1	8392.1	21577	1141.4

Table 25 – Output Stepwise - BIC

Change	Df	Sum of Sq	RSS	AIC
<i>None</i>			13611	1055.0
- density	1	315.6	13926	1055.3
+ wrist	1	200.9	13410	1056.8
+ forearm	1	144.5	13466	1057.9
- ankle	1	476.5	14087	1058.2
- biceps	1	694.7	14306	1062.1
- height	1	1071.4	14682	1068.6
- age	1	1956.9	15568	1083.4
- neck	1	2019.8	15631	1084.4
- knee	1	5757.0	19368	1138.4
- chest	1	8672.4	22283	1173.7

4.4.5 Exhaustive search

This selection method essentially analyzes all possible combinations of explanatory variables to build models through exhaustive search. For each generated model, it evaluates them based on certain selection criteria to assess the quality of the fit. Some of the criteria include AIC (Akaike Information Criteria), BIC (Bayesian Information Criteria), Cp (Mallows Criteria), R-squared, adjusted R-squared, or other statistical tests.

Listing 4.9 shows the R code used to obtain the models, while Table 26 displays all ten generated models. For each model, it is possible to see if the variable was included, denoted by the letter "T", or not, denoted by "F". For example, in model number 1, only the variable "chest" was included in the model; whereas, in model 10, all variables were present.

Listing 4.9 – Exhaustive

```
1 R input :
```

```

2 exhaustive <- regsubsets(weight ~ neck + chest + knee + ankle + biceps +
  forearm + wrist + age + density + height, data = data2, method = "
  exhaustive", nvmax = 10)
3
4 s1 <- summary(exhaustive, matrix.logical=TRUE)
5 s1

```

Table 26 – Exhaustive search

	neck	chest	knee	ankle	biceps	forearm	wrist	age	density	height
1	F	T	F	F	F	F	F	F	F	F
2	F	T	T	F	F	F	F	F	F	F
3	T	T	T	F	F	F	F	F	F	F
4	T	T	T	F	F	F	F	T	F	F
5	T	T	T	F	F	F	F	T	F	T
6	T	T	T	F	T	F	F	T	F	T
7	T	T	T	T	T	F	F	T	F	T
8	T	T	T	T	T	F	F	T	T	T
9	T	T	T	T	T	F	T	T	T	T
10	T	T	T	T	T	T	T	T	T	T

To classify and choose the best model among the generated models, three indicators were extracted from R, and the code is presented in Listing 4.10. These indicators include BIC, which has been used in previous cases, Adjusted R-Squared, which can be a useful indicator for comparing models with different numbers of variables, and Mallows CP. Models with smaller CP values are preferred, as they indicate a better balance between parsimony and predictive capability.

Table 27 summarizes the conclusions of this method and compiles the output of the three indicators for each model. The best model for each criterion is highlighted in bold. When considering Adjusted R-Squared and Mallows CP, the optimal model is number 10, which incorporates all variables. However, based on the BIC criterion, the preferable model is again the one with only eight variables, excluding "forearm" and "wrist" (model number 8).

Listing 4.10 – Exhaustive - indicadores

```

1 R input:
2 s1$adjr2 ### Adjusted R Squared
3 s1$cp ### Mallows CP
4 s1$bic ### BIC

```

Table 27 – Model Indicators

Model	Adjusted R^2	Mallows CP	BIC
1	0.7988	546.2161	-393.9867
2	0.8900	186.5866	-541.5657
3	0.9085	114.4432	-583.4153
4	0.9243	53.1442	-626.8511
5	0.9285	37.5317	-636.8269
6	0.9321	24.5359	-645.2954
7	0.9339	18.5559	-647.5446
8	0.9352	14.7867	-647.7921
9	0.9358	13.1143	-646.0102
10	0.9367	11.0000	-644.7466

4.4.6 Lasso

The next method attempted for selecting the optimal set of variables in this example was Lasso regression. Initially, the response variable was defined, as shown in Listing 4.11 (line 2), along with the covariates (line 3). Subsequently, the best lambda value was determined to penalize the coefficients (line 5 - 7), and the model was fitted accordingly (line 9). The coefficients were then printed to inspect the model (line 13).

Given that the data used in the cross-validation process is random, the optimal lambda value may vary if the process is repeated multiple times. Therefore, the process was repeated fifty times to observe if the selected variables remained consistent across different lambda values. It was observed that only in two out of fifty trials, the selected variables differed from the others.

In the majority of trials (96%), the variable 'forearm' was consistently excluded from the model, as evidenced by the absence of its coefficient in Table 28. However, in the remaining trials, all variables were deemed significant for the model.

Listing 4.11 – Lasso

```

1 R input:
2 Y <- data2$weight
3 X <- data2[, c("neck", "chest", "knee", "ankle", "biceps", "forearm", "
  wrist", "density", "age", "height")]
4
5 cv <- cv.glmnet(as.matrix(X), Y, alpha = 1)
6 best_lambda <- cv$lambda.min
7 best_lambda
8
9 model_lasso <- glmnet(X, Y, alpha = 1, lambda = best_lambda)
10 print(model_lasso)
11

```

```

12 coefficients <- coef(model_lasso)
13 print(coefficients)

```

Table 28 – Coefficients of the Lasso Model

Variable	Coefficient
(Intercept)	-207.1562328
neck	1.8620099
chest	1.4804532
knee	3.3770313
ankle	0.8170034
biceps	0.9173925
forearm	.
wrist	0.9938564
density	-51.2860153
age	-0.1997753
height	0.4598149

4.4.7 SSGL

The last method applied to the example were SSGL. Similarly to simulation, the `glmnet` function is employed, with `lambda1` set at a fixed value of 1 and `lambda0` determined through cross-validation within the range from 1 to 100, following the guidelines provided by Bai et al. (2020). The code used are shown in Listing 4.12.

Listing 4.12 – SSGL

```

1 data2 <- read.csv("data_gordura.csv", header = TRUE, dec = ",", sep = ";")
2
3 X <- data2[, c("neck", "chest", "knee", "ankle", "biceps", "forearm", "
   wrist", "density", "age", "height")]
4 Y <- data2$weight
5 X <- as.matrix(X)
6
7 groups <- 1:ncol(X)
8
9 n <- nrow(data2)
10 train_index <- sample(1:n, round(0.8*n))
11 test_index <- setdiff(1:n, train_index)
12
13 # training
14 X_train <- X[train_index, ]
15 Y_train <- Y[train_index]
16
17 # test
18 X_test <- X[test_index, ]
19 Y_test <- Y[test_index]

```

```

20
21 # cross validation
22 lambda0seq <- 1
23 lambda0seq <- seq(1, 100, length = 100)
24
25 error_min <- Inf
26 best_lambda <- NULL
27
28 for (lambda0 in lambda0seq) {
29   for (lambda1 in seq(0.1, 1, by = 0.1)) {
30     modSSGL <- SSGL(Y = Y_train, X = X_train, lambda1 = lambda1, lambda0 =
      lambda0, groups = groups,
31                     a = 1, b = ncol(X), M = 10, error = 0.001)
32     error <- mean((Y_test - X_test %*% modSSGL$beta)^2)
33
34     if (error < error_min) {
35       error_min <- error
36       best_lambda <- c(lambda1, lambda0)
37     }
38   }
39 }
40
41 # model
42 mod_gglasso <- SSGL(Y = Y, X = X, lambda1 = best_lambda[1], lambda0 = best_
      lambda[2], groups = groups,
43                     a = 1, b = ncol(X), M = 10, error = 0.001)
44
45 coef <- data.frame(Variavel = colnames(X), Coeficiente = mod_gglasso$beta)
46 print(coef)

```

Table 29 presents the conclusions of this method, compiling the output and showing that the best combination of variables does not exclude any.

Table 29 – Coefficients of the Lasso Model

Variable	Coefficient
(Intercept)	-207.1562328
neck	10.0269270
chest	0.8433704
knee	1.4632968
ankle	0.4456183
biceps	-0.7266101
forearm	-0.7682559
wrist	-1.9684323
density	-68.5460589
age	-0.3595442
height	0.3879313

4.4.8 Final Remarks

Once all the methods were applied, it is time to decide which one to follow, as the outputs differed depending on the criteria and the method used. In this example with multiple variables, when methods yielded more than one criterion, the chosen approach was the one that penalized more severely the inclusion of variables, aiming to simplify the model by selecting fewer variables.

Thus, Table 30 presents a summary of the variables selected (marked with an "X") by each method. In most cases, the variables dropped were "wrist" and "forearm". However, for the Lasso method, the only suggested exclusion was "forearm", and for SSGL, none were excluded.

It is important to connect this conclusion with the result obtained in the simulation. Firstly, based on n and p , the scenarios that best fit the examples are 13, 14 or 15 (Table 1). Considering the parameter of the percentage of significant variables, after running the methods and based on Table 30, at least 80% of the variables are significant to the model. Therefore, the scenario that best fits this example is scenario 15.

Based on the metrics presented in Table 11 the best methods to achieve optimal performance would be Backward and Stepwise for this case. It is important to note that all methods demonstrated good performance in the sensitivity metric. Lastly, the group of variables chosen to fit the model for this example included all variables except "wrist" and "forearm."

Table 30 – Variable Selection by Different Methods

	neck	chest	knee	ankle	biceps	forearm	wrist	density	age	height
Backward	X	X	X	X	X	-	-	X	X	X
Forward	X	X	X	X	X	-	-	X	X	X
Stepwise	X	X	X	X	X	-	-	X	X	X
Exhaustive	X	X	X	X	X	-	-	X	X	X
Lasso	X	X	X	X	X	-	X	X	X	X
SSGL	X	X	X	X	X	X	X	X	X	X

So, the final model is presented in Listing 4.13 with the code used and the R output showing the coefficients for each variable. The p-value of the F-statistic test (line 32) indicates that the model fits the data well according to the defined criteria, and the adjusted R-squared (line 31) also suggests a good fit, approaching 94%.

Listing 4.13 – Model 3

```

1 R input:
2 model3 <- lm(weight ~ neck + chest + knee + ankle + biceps + age + density
  + height, data = data2)
3 summary(model3)
4
```

```

5 R output:
6 > summary(model3)
7
8 Call:
9 lm(formula = weight ~ neck + chest + knee + ankle + biceps +
10     age + density + height, data = data2)
11
12 Residuals:
13      Min       1Q   Median       3Q      Max
14 -19.097   -4.353   -0.177    4.337   45.211
15
16 Coefficients:
17             Estimate Std. Error t value Pr(>|t|)
18 (Intercept) -178.74476    43.49878   -4.109 5.43e-05 ***
19 neck         2.11434     0.35209    6.005 6.93e-09 ***
20 chest        1.49479     0.12013   12.443 < 2e-16 ***
21 knee         3.37620     0.33302   10.138 < 2e-16 ***
22 ankle        1.05610     0.36210    2.917 0.003870 **
23 biceps        0.92442     0.26248    3.522 0.000512 ***
24 age         -0.24452     0.04137   -5.911 1.15e-08 ***
25 density     -86.25503    36.33625   -2.374 0.018384 *
26 height        0.62912     0.14384    4.374 1.81e-05 ***
27 ———
28 Signif. codes:  0      ***      0.001      **      0.01      *      0.05      .      0.1
29                  1
30 Residual standard error: 7.484 on 243 degrees of freedom
31 Multiple R-squared:  0.9372,    Adjusted R-squared:  0.9352
32 F-statistic: 453.4 on 8 and 243 DF,  p-value: < 2.2e-16

```

Concerning residual analysis, tests were conducted to check for normal distribution and constant variance of the errors. For this purpose, the Shapiro-Wilk and Breusch-Pagan tests were employed, along with graphical analysis.

The residuals of the model were plotted in Figure 10 using the *qqnorm* and *qqline* functions. In Figure 10, a noticeable deviation towards the upper side suggests that the residuals do not follow a normal distribution. This observation is supported by the Shapiro-Wilk test, as shown in Listing 4.14 (line 11), where the p-value is smaller than the alpha criterion of 5%. Therefore, the null hypothesis of normality needs to be rejected, indicating that the residuals do not conform to a normal distribution.

Listing 4.14 – Normality test

```

1 R input:
2 qqnorm(residual3, pch=16, cex.lab=1.5, cex=2, cex.axis=1.5, cex.main = 2) ;
   qqline(residual3)
3 shapiro.test(residual3)

```

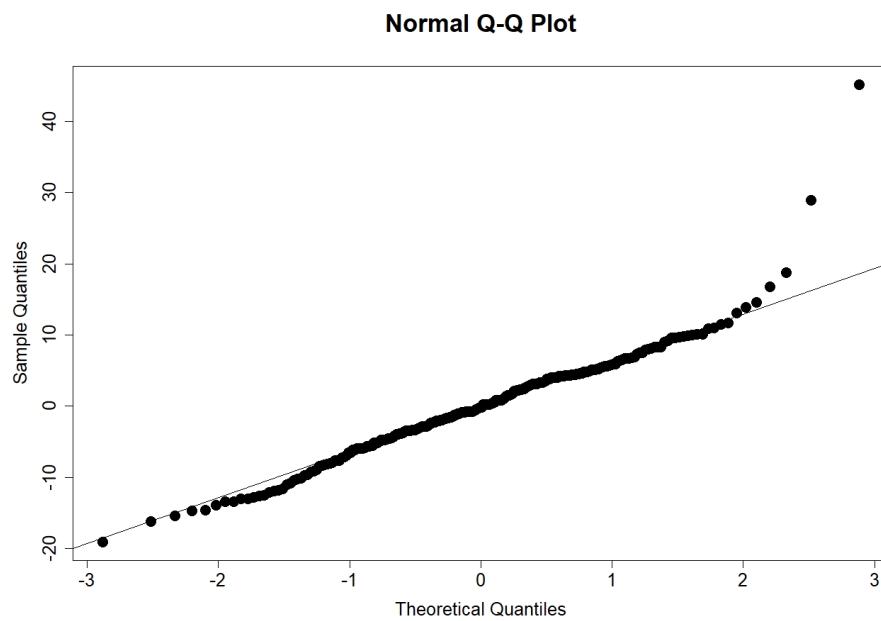


Figure 10 – Normality test - anthropometric data

Source: By the author

```
4
5 R output:
6 > shapiro.test(residual3)  #Teste de normalidade de Shapiro Wilk
7
8     Shapiro-Wilk normality test
9
10 data:  residual3
11 W = 0.94176, p-value = 1.865e-08
```

Observing Figure 11, no discernible pattern is apparent in the distribution of the points, although some are notably distant from others. Additionally, the result of the Breusch-Pagan test in Listing 4.15 (line 12) suggests rejecting the null hypothesis, indicating that the variance of errors is not constant. This implies that the data exhibit heteroscedasticity.

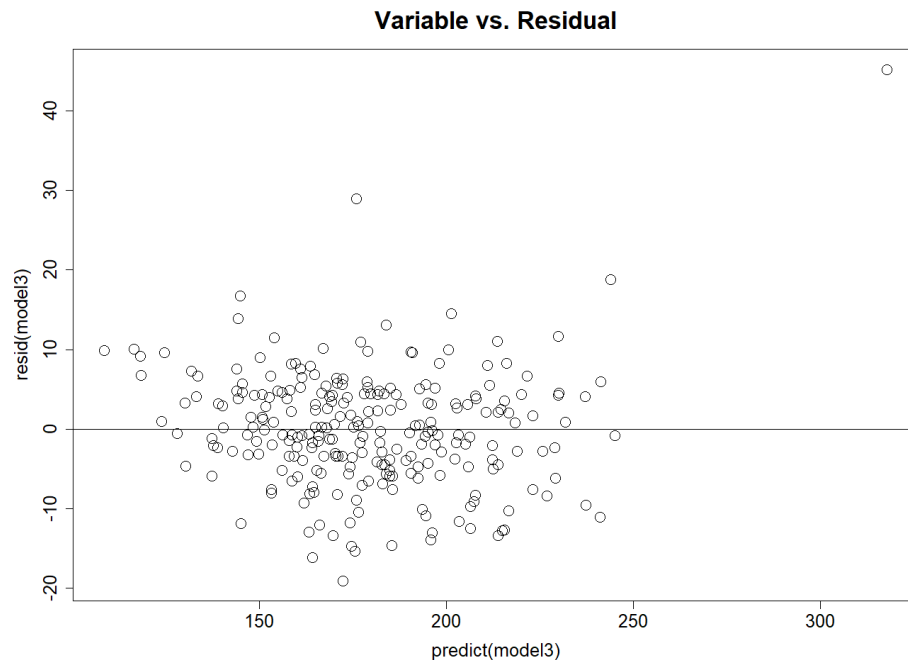


Figure 11 – Variance test - anthropometric data

Source: By the author

Listing 4.15 – Variance test

```

1 R input:
2 plot(predict(model3), resid(model3), main="Variable vs. Residual", cex.lab
   =1.5, cex=2, cex.axis=1.5, cex.main = 2)
3 abline(h=0)
4 bptest(model3)
5
6 R output:
7 > bptest(model3)
8
9      studentized Breusch-Pagan test
10
11 data:  model3
12 BP = 66.059, df = 8, p-value = 2.979e-11

```

So, finally, to transform this data into a usable model, it is important to consider performing a transformation (logarithmic, squared root, Box-Cox), cross-validation, or perhaps examining the data to identify any issues or erroneous data. For instance, Figure 11 illustrates that some points are considerably distant, indicating the need to investigate if there are any problems associated with them.

5 CONCLUSION

Based on the findings from the comparative analysis of variable selection techniques, this study has provided valuable insights into the performance of various methods across different scenarios. Initially, simulated data allowed for a rigorous evaluation of methods such as Backward, Stepwise, Lasso, and SSGL, assessing metrics like accuracy, sensitivity, specificity, and F1 score. The results indicated that traditional methods like Backward and Stepwise generally outperformed in scenarios with smaller or larger sample sizes n , highlighting their strengths and the computational challenges with larger p values. Conversely, newer methods such as SSGL demonstrated superior performance in scenarios with larger numbers of variables p and stronger beta coefficients, showcasing their effectiveness in handling high-dimensional data where precise feature selection is crucial.

Furthermore, Backward and Stepwise methods showed comparable outcomes across diverse datasets. Sensitivity emerged as a consistently high-performing metric, underscoring its importance in correctly identifying relevant variables, albeit subject to variations based on parameter settings. Computational efficiency analysis revealed that Lasso exhibited minimal computational overhead across all scenarios. As affirmed by Tibshirani (1996), the lasso is a versatile and effective tool for regression analysis. Whereas SSGL's performance varied based on beta strength, n , and p sizes. Notably, SSGL represents the state of the art in variable selection methods, offering a balanced approach to feature selection and excelling in high-dimensional data settings while maintaining high precision in selecting valuable variables.

In conclusion, this research enhances understanding of variable selection methods' adaptability and performance under varying data characteristics and computational constraints. It emphasizes the critical role of methodological choices in achieving optimal model outcomes, particularly in modern data analytics contexts characterized by expanding dataset scale and complexity.

Moving forward, leveraging advancements in methods like SSGL will be crucial in advancing the state of the art in predictive modeling and decision support systems, ensuring alignment with evolving data science practices and demands. Future directions could involve expanding analyses to encompass higher levels of n and p sizes beyond those explored here, utilizing more robust computational resources. Additionally, exploring additional scenarios such as p exceeding n , incorporating other methods in simulations (e.g., forward selection, exhaustive search, random forest), varying beta strengths more comprehensively, and testing non-linear models could further enrich and extend the findings of this study.

6 BIBLIOGRAPHY

Bai, R., Moran, G. E., Antonelli, J. L., Chen, Y., and Boland, M. R. (2020). Spike-and-Slab Group Lassos for Grouped Regression and Sparse Generalized Additive Models. Retrieved from March 4, 2020.

R Core Team (2024). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL <https://www.R-project.org/>.

Ročková, V., and George, E. I. (2018). The Spike-and-Slab LASSO. *Journal of the American Statistical Association*, 113(521), 431-444. <https://doi.org/10.1080/01621459.2016.1260469>

Tibshirani, R. (1994). Regression Shrinkage and Selection via the LASSO. *Journal of the Royal Statistical Society, Series B*, 58, 267–288.

Google Colaboratory (2024). Colaboratory: Free Jupyter Notebook Environment. Available at: <https://colab.research.google.com/>.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning*.

Python Software Foundation. Python Language Reference, version 3.10. Available at: <https://www.python.org/>

Witten, I. H., Frank, E., and Hall, M. A. (2011). *Data Mining: Practical Machine Learning Tools and Techniques* (3rd ed.). Morgan Kaufmann.