BACHELOR'S DEGREE THESIS

# Degree in Statistics

**Title:** Proximal Algorithms: ISTA and FISTA for L1-Regularized Regression

**Author:** YingHong Chen

**Advisor:** Esteban Vegas Lozano; Ferran Reverter Comes

**Department:** Genetics, Microbiology and Statistics, University of Barcelona

**Academic year:** 2024-2025

UNIVERSITAT DE BARCELONA

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat de Matemàtiques i Estadística

## Abstract

Linear regression models are widely used across fileds like medicine, biology, and economics. This work explores the use of proximal gradient methods, particularly the ISTA and its accelerated version, FISTA, which are simple and efficient algorithms for solving optimization problems with non-differentialble penalties such as L1-norm used in Lasso regression.
A package called `ProxReg` was made to make it easier to use the algorithms. It supports prediction and classification tasks with binary, numeric and multinomial target variables using Lasso regression model. And it also includes Ridge, OLS regression, cross-validation tools, and image reconstruction features. The efficacy and performance of the proposed proximal gradient methods are evaluated by comparing them with the Lasso regression results based on the `glmnet` package coordinate descent method, using real-world and simulated data.



**Keywords:** Linear Regression; Machine Learning; Optimization; Proximal Operator; L1-Regularized Regression; R package



## Resumen

Los modelos de regresión lineal se utilizan ampliamente en diversas áreas como la medicina, la biología y la economía. Este trabajo explora el uso de métodos de gradiente proximal, en particular el algoritmo ISTA y su versión acelerada FISTA, diseñados para resolver problemas de optimización con penalización no diferenciables, como la norma L1 en la regresión Lasso.
Para el estudio, se ha desarrollado un paquete `ProxReg` para facilitar el uso de estos algoritmos. Esta permite realizar tareas de predicción o clasificación para variables binarias, numéricas y multinomiales. También incluye funciones para regresión Ridge, regresión por mínimos cuadrados ordinarios, métodos de validación cruzada y reconstrucción de imágenes. La eficacia y el rendimiento de los métodos propuestos se evalúan comparándolos con los resultados de la regresión Lasso del paquete `glmnet`, utilizando datos reales y simulados.



**Palabras clave:** Regresión lineal; Aprendizaje automático; Optimización; Operador proximal; Regresión regularizada con L1; Paquete R

# AMS Classification

- 62J07 Ridge regression; shrinkage estimators
- 62-07 Data analysis
- 65K10 Optimization and variational techniques
- 90C25 Convex programming

# Acknowledgements

Firstly, I would like to express my sincere thanks to my tutors for proposing this research topic and for guiding me through the development of this thesis. Their ongoing support, feedback, and suggestions were fundamental in helping me explore and understand the practical implementation of Lasso regression and proximal optimization techniques such as ISTA and FISTA.

Secondly, I am also grateful to my friends for their companionship throughout this journey. Your support during the most challenging moments has meant a great deal to me.

Lastly, I want to thank my university and the professors who contributed to my academic formation. Their dedication and guidance helped me discover my interest in this field, and I look forward to continuing to grow and contribute within it in the future.

# Contents

# List of Figures

# List of Tables

# Glossary

| Term | Description |
|---|---|
| **Linear Regression** | A model that estimates the linear relationship between a dependent variable and one or more independent variables. |
| **Ordinary Least Squares Regression** | Linear regression that estimates coefficients by minimizing the sum of squared residuals. |
| **Lasso Regression** | A linear regression method that uses L1 regularization to shrink some coefficients to exactly zero, it allows feature selection and reduces model complexity and multicollinearity. |
| **Logistic Lasso Regression** | Lasso regression used for binary classification tasks using a logistic loss function. |
| **Multiclass Lasso Regression** | Lasso regression used for classification tasks with more than two classes using the softmax function. |
| **Ridge Regression** | A linear regression method that uses L2 regularization to penalize large coefficients, it reduces multicollinearity without setting any coefficient to zero. |
| **Multicollinearity** | A condition where two or more predictors are highly correlated with each other, making coefficient estimation unstable. |
| **ISTA** | Optimization algorithm for solving Lasso regression problems using a proximal operator, it has a convergence rate of $O(\frac{1}{k})$. |
| **FISTA** | An accelerated version of ISTA with a convergence rate of $O(\frac{1}{k^2})$. |

| Term | Description |
|---|---|
| **Proximal Operator** | Mathematical tool used for solving optimization problems with non-differentiable objective functions. |
| **Regularization** | Technique used in optimization problems to prevent overfitting |
| **Coordinate Gradient Descent Algorithm (CGDA)** | An optimization method that minimizes objective function by updating one parameter at a time while holding others fixed. |
| **Smooth L1 Algorithm (SLA)** | A technique that approximates to the L1 norm with a smooth differentiable function. |
| **Path Following Algorithm (PFA)** | An algorithm that traces the solution path of Lasso as the regularization parameter changes. |
| **Sigmoid Function** | A mathematical function used in logistic regression that transforms real numbers to a value between 0 and 1. |
| **Backtracking Line Search** | Method that finds a proper step size by starting large and repeatedly shrinking until a sufficient reduction in the function value is achieved. |
| **Evaluation Metrics** | Measures used to assess the performance of models in prediction and classification tasks. |
| **Mean Absolute Error (MAE)** | Average absolute difference between predicted and true values. |
| **Root Mean Squared Error (MAPE)** | Average absolute percent error between predicted and true values. |
| **Coefficient of Determination ($R^2$)** | Percentage of variance explained by the model. |

| Term | Description |
|---|---|
| **Adjusted Coefficient of Determination** | Percentage of variance explained by the model adjusted by the number of predictors. |
| **Confusion Matrix** | A table that shows correct and incorrect classifications |
| **ROC Curve** | A graphic that shows true positive rate against false positive rate. |
| **Area Under the Curve (AUC)** | The area under the ROC curve, reflects the model's ability to distinguish between classes. |
| **True Positive (TP)** | Correctly predicted positive cases. |
| **True Negatives (TN)** | Correctly predicted negative cases. |
| **False Positives (FP)** | Incorrectly predicted positive cases. |
| **False Negatives (FN)** | Incorrectly predicted negative cases. |
| **Accuracy** | The proportion of correctly classified instances among all observations. |
| **Sensitivity** | The proportion of actual positives correctly identified. |
| **Overfitting** | A condition when the model learns training data too well, which makes it perform poorly on unseen data. |
| **Hyperparameter** | A set of parameters before model training and affects the model's learning behavior. |
| **Feature Selection Consistency** | The stability of features selected by models across various runs on the same dataset. |

# Notations

| Term | Description |
|---|---|
| $R^n$ | $n$-dimensional real-valued vector space |
| $x, X$ | Input feature (vector and matrix) |
| $y$ | Output or target variable |
| $\hat{y}$ | Predicted value |
| $\beta$ | Regression coefficient (scalar and vector) |
| $\lambda$ | Regularization parameter |
| $\theta$ | Logistic regression parameters |
| $L()$ | Loss function with respect to parameters |
| $\nabla$ | Gradient operator |
| $\mathcal{L}_1, \mathcal{L}_2$ | L1 and L2 regularization terms |
| $\|\cdot\|_1$ | L1 norm (sum of absolute values) |
| $\|\cdot\|_2$ | L2 norm (Euclidean norm) |
| $\text{prox}_\lambda(z)$ | Proximal operator for regularization term $\lambda$ |
| $S(\cdot)$ | Shrinkage function |
| $A^T, A'$ | Transpose of matrix $A$ |
| $\epsilon$ | Tolerance value or error |
| $F(x), \hat{F}(x)$ | Objective function and its estimate |
| $\sigma_{max}(X)$ | Largest singular value of a matrix $X$ |

# Chapter 1

# Introduction

## 1.1 Context and Motivation

In 1805, the renowned French mathematician Adrien-Marie Legendre published in his article the method of least squares as a new technique for determining the cometary orbits in astronomy. Later, Carl Friedrich Gauss refined this approach by combining it with principles of probability and normal distribution, which later became fundamental assumptions in linear regression.

The concept of regression analysis dates back to Francis Galton, a prominent British scientist known for his important contributions across biology, mathematics, and psychology. Influenced by the work of his cousin, David Darwin, Galton's study of sweet pea seed heredity [11] led him to be the first person to describe the phenomenon of "regression toward the mean", where he observed that the sizes of offspring tended to move closer to the average size of the population rather than replicating the extreme patterns of their parents. In 1886, Galton extended this concept through experiments on human height and plotted the famous diagram representing what he termed "regression towards mediocrity" [12].

In the early 1890s, Karl Pearson advanced the field by introducing the correlation coefficient and bivariate regression slope [23]. Over the following years, building on the work of his predecessors, Ronald Fisher later made a significant contribution to the field by generalizing the simple regression model into multiple regression [9], enabling the analysis of relationships among multiple variables. In the late 20th century, Ridge and Lasso regression were introduced to handle the multicollinearity problem in high-dimensional data [18] [26], which established themselves as foundational techniques in modern statistics and machine learning.

In today's data-driven world, regression models are essential for uncovering relationships and

making predictions and classifications in different domains such as biology, finance, and public health. Linear regression, especially Lasso regression, has become relevant for analyzing complex, high-dimensional datasets. Meanwhile, proficiency in statistical programming tools like R has become indispensable for implementing these models in applied research.

However, the diversity of statistical packages can make implementation tedious and time-consuming. There are several R packages for Lasso or regularized regression, such as `glmnet` and `lars`, that rely on coordinate descent or least angle regression. Among them, `glmnet` remains the most widely used due to its speed, stability, and suitability for large datasets.

However, these algorithms are not the only approaches for solving Lasso optimization problems. Proximal gradient methods like the Iterative Shrinkage Thresholding Algorithms (ISTA) and its faster version (FISTA) present better convergence rates and flexibility in handling non-differentiable terms like the L1 penalty.

To address these gaps, an R package, `ProxReg` has been developed. It provides implementations of both ISTA and FISTA, and supports regression and classification for various types of response variables, including continuous, binary, and multinomial outcomes. Additionally, it integrates visualization tools, cross-validation, and even image reconstruction using sparse regression. In addition to Lasso, the package also includes implementations of Ordinary Least Squares (OLS) and Ridge regression, to provide a complete suite of linear modeling tools for both low- and high-dimensional data. Moreover, preliminary results of the package were presented at the workshop held for the 10th Anniversary of GRBIO on January 30, where its features and potential applications were explained to an audience of biostatisticians, bioinformaticians, and applied researchers.

## 1.2 Objectives

The primary objective is to develop, implement, and compare some widely used linear models in the machine learning domain, with a particular focus on the development of the algorithms, such as ISTA and FISTA, for L1 regularized regression models using proximal operator methods.

To support this objective, the study carried out several tasks, such as:

- The search and selection of suitable datasets for model evaluation
- The development of code for algorithm implementation
- The creation of an R package
- The submission of the package to CRAN for public use

As a secondary objective, the study presents the design and development of the `ProxReg` R package. This package serves as a practical framework for fitting linear models, implementing algorithms, and applying model evaluation techniques. It supports reproducibility and allows benchmarking of the implemented models, and facilitates the approach to the main objective.

Finally, the study includes practical application of ISTA and FISTA through different case studies, discusses the strengths and limitations of the algorithms, and offers insights into the scalability and performance of the package in high-dimensional settings.

# Chapter 2

# Methodology

## 2.1 Ordinary Least Squares Regression

The ordinary least squares regression (OLS) [5] has proved to be one of the most used regression models because of its simplicity. One benefit of using the OLS model is that it gives unbiased coefficient estimators, meaning that $E(\hat{\beta}) = \beta$. These are found by using the least squares method, which minimizes the residual sum of squares (RSS) between the observed value $(y_i)$ and the predicted values $(\hat{y_i})$ of the response variable. This is equation takes the following form in math:

$$RSS = \sum_{i}^{N} (y_i - \hat{y_i})^2$$

However, the OLS model is not always suitable for all types of data. To ensure the validity of the model for further data analysis and prediction, it must meet certain conditions outlined by the Gauss-Markov theorem for the OLS estimates to be the best (or most precise) linear unbiased estimators, which requires linearity between the predictors and the response variable, homoscedasticity and errors following normal distribution.

## 2.2 Ridge Regression

In many situations, databases used for linear modeling are highly dimensional, containing a large number of predictors. A large number of predictors can increase the accuracy of the model, but it also causes a serious multicollinearity issue [20]. This phenomenon occurs when the predictors are highly correlated with each other and the matrix $X'X$ is almost singular, therefore, the OLS estimators will have large variance and low accuracy. The most used metrics to detect the multicollinearity are the Variance Inflation Factor (VIF) for each predictor or

the condition number (k) of the regression matrix X. When $VIF > 10$ or $k > 30$, a serious problem of multicollinearity is generally considered.

To address multicollinearity, one popular approach is variable selection, which involves the use of stepwise selection [31] to choose the most relevant variables for the model with the lowest Akaike Information Criterion (AIC). Partial Least Regression (PLS) [13] and Principal Components Regression (PCR) [1] are two other options. These methods take all the original variables and combine them in a linear way to make new, independent components. However, both techniques have their limitations. For instance, stepwise selection may not be reliable when there are more predictors than observations. Moreover, interpreting the coefficients from PCR and PLS can be challenging, since both methods use new variables based on all the explanatory variables, therefore, the model simplicity is not approached.

Other techniques for dealing with multicollinearity are Ridge and Lasso regression. The Ridge [21] estimates minimize the sum of squared residuals plus a penalty proportional to the squares of the coefficients. The penalty term $\lambda$ increases the eigenvalues of the matrix $X'X$ so that it reduces the variance of the coefficients and mitigates the impact of multicollinearity. The equation minimized by Ridge's regression is:

$$\min_{\beta} \sum_{i=1}^{N} (y_i - \sum_{j=1}^{p} \beta_j x_{ij})^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

Where $p$ presents the number of predictors and $N$, the number of observations. This regularization approach restricts the coefficients, making the least relevant variables have the values of coefficients proximate to 0. However, if the purpose is centered on feature selection, ridge regression might not be effective since any of the variables are eliminated or have coefficients precisely at 0. Therefore, Lasso regression would be a solution to the issue.

## 2.3 Lasso Regression

The Lasso regression [26] introduced by Professor Robert Tibshirani at Stanford University, is a regularization technique used to improve the performance of linear models. Like Ridge regression, Lasso also minimizes the residual sum of squares. However, the key difference between the methods lies in their regularization approaches: Ridge regression adds a penalty based on the sum of squared coefficients (L2 norm), while Lasso regression adds a penalty based on the sum of absolute values (L1 norm) [2], which can reduce some coefficients to an exact zero value and makes Lasso especially useful for feature selection. In such a way, only

the most important variables are kept in the model, making it suitable for high-dimensional data and a useful tool to handle the multicollinearity problem.

Figure 2.1 presents the geometric interpretation of Ridge and Lasso [16]. The ellipses are the contours of the residual sum of squares and the shaded region is the constraint imposed by the regularization term. In Ridge regression (right), the constraint is circular and tends to shrink coefficients to zero but rarely sets them exactly to zero. On the other hand, Lasso regression (left) has a diamond-shaped constraint, where the ellipse contacts with the corners making making the coefficients shrink to zero.

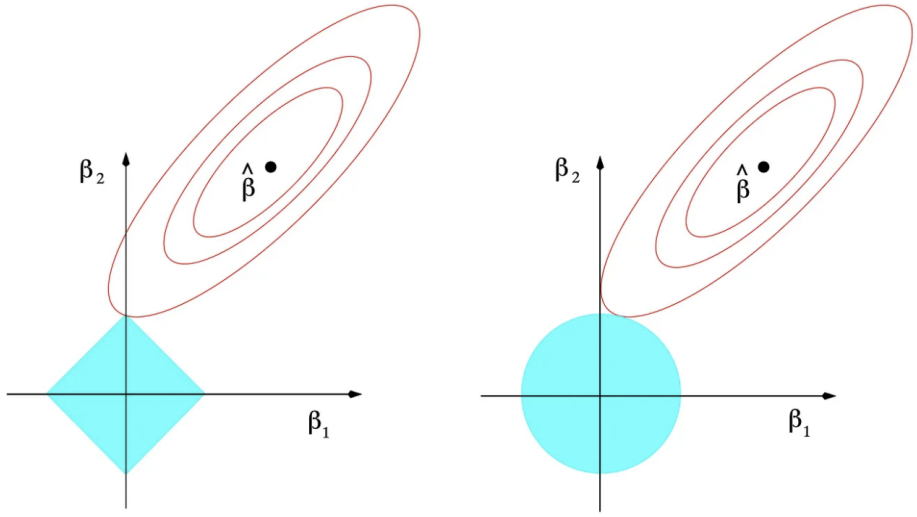


Figure 2.1: Geometric interpretation for Lasso and Ridge

Mathematically, consider $y_i$ and $x_{ij}$ as responses and regressors for the i-th observation, where $i = 1 \dots N$ and $j = 1...p$, Lasso regression solves the following optimization problem:

$$\min_{\beta} \sum_{i=1}^{N} (y_i - \sum_{j}^{p} \beta_j x_{ij})^2 + \lambda \sum_{j}^{p} |\beta_j|$$

Or equally minimizes the sum of squares to the constraint that the total absolute value of the coefficients is less than or equal to a constant.

$$\min_{\beta} \sum_{i=1}^{N} (y_i - \sum_{j}^{p} \beta_j x_{ij})^2 \, subject \, to \sum_{j}^{p} |\beta_j| \leq t$$

Where $\lambda$ is a regularization parameter that controls the strength of the penalty and $t$ is a fixed bound on the total absolute value of the coefficients.

### Logistic Lasso Regression

Lasso regression can be adapted for classification problems where the outcome is categorical rather than numeric [30]. A common example is binary classification, such as cancer diagnosis, where the dataset often consists of many features reflecting the patient's condition, like age, gender, and tumor-related measurements. In this context, cancer statuses are usually binary qualitative variables labeled as benign and malignant. Therefore, the traditional Lasso regression for continuous response variables is no longer appropriate, and Lasso is applied to logistic regression, which estimates the probability of a class using a sigmoid function.

**Sigmoid Function:**

$$h_\theta(x_i) = \frac{1}{1 + e^{-\theta_i x_i}}$$

The loss function becomes the logistic loss that measures how well the predicted probabilities match the true labels.

**Logistic loss:**

$$L(\theta) = -\frac{1}{N}\sum_{i=1}^{N} y_i log(h_\theta(x_i)) + (1 - y_i) log(1 - h_\theta(x_i)) + \lambda \sum_{j=1}^{p} |\theta_j|$$

### Multiclass Lasso Regression

A multiclass classification problem is a generalization of binary classification, where the response variable is categorical with more than two classes. For example, one might predict the stage of a disease or classify types of flowers based on various features.

In multiclass classification problems, the sigmoid function used in logistic regression is generalized by the softmax function that transforms a vector of unnormalized log probabilities $(z_i)$ into a probability distribution over all classes $(C)$.

**Softmax Function**

$$S(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}}$$

There $z_i$ presents the log probabilities for each class, and *C* is the number of classes. The softmax function ensures that the predicted probabilities for all classes are positive and sum to 1. The model is trained by minimizing the cross-entropy loss that measures the discrepancy between the predicted probabilities and true class labels. The Lasso multinomial loss function is :

$$L(\beta) = -\frac{1}{N}\sum_{i=1}^{N} log(S(z_i)_{y_i}) + \lambda \sum_{j=1}^{p} |\beta_j|$$

## 2.4 Algorithms for Lasso Regression Implementation

There are several algorithms [32] to optimize the objective function of Lasso regression, expressed mathematically as:

$$F(\beta) = \frac{1}{2n}||Y - X\beta||_2^2 + \lambda||\beta||_1$$

Where:

- $Y$ is the vector of observed values
- $X$ is the matrix of predictors
- $\beta$ represents the coefficients, including intercept
- $\lambda$ is the regularization parameter

The commonly used methods include the coordinate gradient descent algorithm (CGDA), smooth L1 algorithm (SLA), path following algorithm (PFA), iterative shrinkage threshold algorithm (ISTA), and its accelerated version, the fast iterative shrinkage threshold algorithm (FISTA).

Table 2.4 compares the convergence rates of the algorithms, among those with established theoretical convergence rates, FISTA presents the fastest rate.

| **Algorithm** | **Convergence rate** |
|---|---|
| Coordinate Descent | $\mathcal{O}(\frac{1}{k})$ |
| ISTA | $\mathcal{O}(\frac{1}{k})$ |
| FISTA | $\mathcal{O}(\frac{1}{k^2})$ |
| Smooth-L1 | $\mathcal{O}(\frac{1}{k})$ |
| Path-following | Not guaranteed |

Table 2.1: Convergence rates of common Lasso algorithms

### 2.4.1 Coordinate Descent Algorithm

The coordinate descent algorithm [29] is a widely used method for solving the Lasso optimization problem, particularly in R packages such as `glmnet`. Unlike other commonly used algorithms that update simultaneously all variables, the method chooses one variable or coordinate at a time while keeping all other variables fixed. The algorithm then optimizes the chosen variables based on the current values of the others and repeats the process for the remaining variables until convergence is achieved. This iteration process continues until the algorithm reaches a solution that minimizes the objective function.

The method is simple to implement and computationally efficient, particularly for high-dimensional problems where the number of predictors is large. The convergence rate of coordinate descent is generally $O(\frac{1}{k})$, where $k$ represents the number of iterations.

### 2.4.2 Smooth L1 Algorithm

The Smooth L1 Algorithm (SLA) is another method used for the Lasso optimization problem and has a convergence rate of $O(\frac{1}{k})$. Unlike coordinate descent, which minimizes directly the objective function, non-differentiable at zero due to the L1-norm term, the SLA introduces a **surrogate function** [24] that approximates the L1 norm in a way that it is differentiable and smooth at zero.

Mathematically, the surrogate function is presented as:

$$|x| \approx \frac{1}{\alpha}[log(1 + exp(-\alpha x)) + log(1 + exp(\alpha x))] = \phi_\alpha(x)$$

Here $x$ is the coefficient in the Lasso regression and $\alpha$ is a hyperparameter that controls how closely the surrogate function ($\phi_\alpha(x)$) approximates the $|x|$. As the $\alpha$ value increases, the surrogate function becomes closer to the original L1 norm.

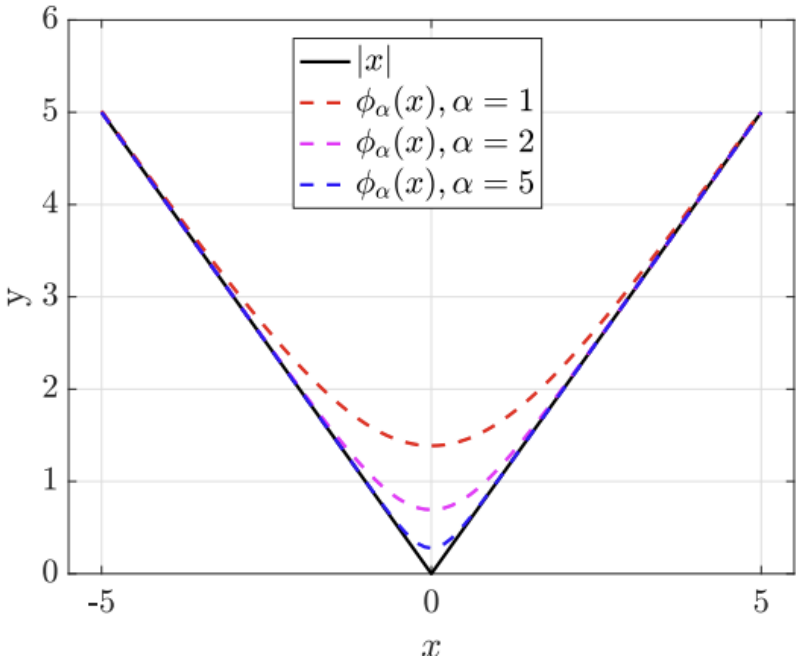


Figure 2.2: Approximation to |x| in smooth L1 algorithm,

Consequently, the objective function for Lasso regression with the L1-norm term replaced by the surrogate function becomes:

$$F_\alpha(\beta) = \frac{1}{2n}||Y - X\beta||_2^2 + \lambda \sum_i^p \phi_\alpha(\beta_i)$$

### 2.4.3 Path Following Algorithm

The Path Following Algorithm (PFA) [10] aims to find the optimal solution for Lasso regression problem by following the path of possible solutions as the regularization parameter $\lambda$ changes.

The algorithm starts by choosing a large value, $\lambda_0$ which forces all the estimated coefficients to be zero. As $\lambda$ decreases, the model gradually includes more variables into the support set$S(\beta)$, which contains predictors with non-zero coefficients. Moreover, the support set has the property of being unchanged between certain intervals of$\lambda$, and only changes when it reaches a kink point where the model decides to include some variables.

Once the support set is identified, PFA estimates the coefficients for the variables included in $S(\lambda)$. Instead of recalculating the solution for every $\lambda$. The algorithm updates the coefficients based on the correlation between the previously estimated coefficients and the new ones as the support set evolves.

However, PFA presents some limitations. Firstly, its convergence rate is not guaranteed, and it can fail to find solutions in cases when the design matrix $X$ presents independencies between predictors. Especially when the structure of the predictors causes difficulties in following the path, PFA might struggle to identify correctly the true support set and fail to converge to the optimal solution.

## 2.5 ISTA and FISTA Algorithms

The lasso regression is a powerful method for multicollinearity and feature selection in high-dimensional data. However, its objective function is non-differentiable due to the L1 penalty term, which complicates direct optimization using derivatives, as in the case of ridge regression, which has a differentiable loss function. Over the years, researchers have developed a wide range of algorithms to approximate the solution for Lasso regression. One of the most widely used methods is coordinate descent, which is also implemented in the `glmnet` package for solving lasso regression. The method minimizes iteratively the loss function by optimizing each coordinate at a time while keeping other parameters fixed, based on the gradient of the differentiable part of the objective.

The `ProxReg` package includes methods that use proximal gradient techniques, particularly the Iterative Shrinkage-Thresholding Algorithm (ISTA) and its faster version, the Fast Iterative Shrinkage-Thresholding Algorithm (FISTA). These methods use proximal operators to solve the Lasso regression problem.

### 2.5.1 The Iterative Shrinkage Thresholding Algorithm (ISTA)

The Iterative Shrinkage-Thresholding Algorithms (ISTA) [14], is one of the most widely used iterative optimization techniques for Lasso regression, renowned for its simple implementation and efficiency. It has a convergence rate of $O(\frac{1}{k})$ where *k* represents the iteration counter.

Consider the Lasso objective function :

$$F(\beta) = \underbrace{\frac{1}{2n}\|Y - X\beta\|_2^2}_{g(\beta)} + \underbrace{\lambda\|\beta\|_1}_{h(\beta)}$$

Where the first term is differentiable and the second is non-differentiable due to the $L_1$-norm regularization.

The ISTA algorithm with a fixed step aims to approximate the minimum point of such an optimization function using a proximal gradient update. The gradient of the smooth part or the differentiable term is given by:

$$\nabla g(\beta) = -\frac{1}{N}X'(Y - X\beta)$$

The update rule at iteration k is:

$$\beta^{k+1} = S_{\lambda t}(\beta^k - \frac{1}{L}\nabla g(\beta^k))$$

Where $L$ is the Lipschitz constant calculated as the largest eigenvalue of $X'X$, $L = \lambda_{max}(X'X)$, and $S_{\lambda t}()$ is the thresholding operator with the condition:

$$S_{\lambda t}(\beta_i) = \begin{cases} \beta_i - \lambda t & \text{if } \beta_i > \lambda t, \\ 0 & \text{if } -\lambda t \leq \beta_i \leq \lambda t, \\ \beta_i + \lambda t & \text{if } \beta_i < -\lambda t \end{cases}$$

Occasionally, it is difficult to determine the appropriate fixed step size, and the convergence may be slow if the step size is constant. To solve this issue, the alternative solution is to use the technique of *backtracking line search* [8] that automatically adjusts the step size based on the proximity to the minimum point. Once the proximal gradient ($\nabla g(\beta^k)$) is obtained, the initial step is set to 1, and the Lasso cost function $F(\beta^k)$ is evaluated. If the condition $F(\beta^{k+1}) \leq F(\beta^{k-1})$ is violated, it means that the step is too large and has surpassed the minimum. Hence, the step size *t* is reduced by multiplying it with a constant *c* ranging between 0 and 1 ($0 < c < 1$) to ensure the next iteration reaches the minimum more effectively.

Both techniques of the algorithm, fixed step size and backtracking line search, begin with an initial vector of $\beta^0$, and use the ISTA algorithm to iteratively update the coefficient values until

the convergence is reached by reaching the maximum number of iterations or obtaining a convergence error lower than a predetermined tolerance.

In Algorithm 1, an outline of the ISTA algorithm is presented. Where the inputs are a response variable $Y$, a matrix of predictors $X$, the regularized parameter $\lambda$, the maximum number of iterations $k$, and the tolerance $Tol$. Once the inputs are set, it starts by initializing the coefficients and calculating the Lipschitz constant as the largest eigenvalue of the matrix $X'X$, $L = \sigma_{max}(A'A)$ which is used to determine the step as $1/L$. In each iteration, the gradient is calculated, and the coefficients are updated using the proximal operator. The process repeats until the maximum number of iterations is reached or the convergence error falls below the specified tolerance.

**Algorithm 1** : ISTA with fixed step

**Inputs:**
$Y$ , $X$ , $\lambda$ ,
$K$
Tol
**Initialize:**
$\beta_0$ (initial coefficients)
$L =$ Lipschitz constant
race $= 1/L$
**for** $i = 1$ to $K$ **do**
    Compute Gradient : $\nabla g = X'(Y - X\beta_{i-1})/N$
    Update: $\beta_i = S_{\lambda,\text{race}}(\beta_{i-1} - \text{race} \times \nabla g)$
    Compute Error
    **if** Error $<$ Tol **then**
        **break**
    **end if**
**end for**

ISTA with a fixed step size can be slow to converge, especially when the Lipschitz constant $L$ is large, which makes the size small. Backtracking line search offers a dynamic and adaptive step that ensures faster convergence and stability. Consider $g(\cdot)$ the smooth part of the Lasso objective function $F(\beta)$, $h(\cdot)$ the non-differentiable part, and $G_t(\beta)$ the generalized gradient. The backtracking line search updates the step size in every iteration when:

$$g(\beta - tG_t(\beta)) > g(\beta) - t\nabla g(\beta)^t G_t(x) + \frac{t}{2}||G_t(\beta)||_2^2$$

**Algorithm 2** :ISTA with Backtracking Line Search

**Inputs:** $Y$, $X$, $\lambda$, $K$, Tol

**Initialize:**

$\beta_0$ (initial coefficients)

$L =$ Lipschitz constant

$t = 4/L$ (initial step size)

$\eta \in (0, 1)$ (shrink factor)

**for** $i = 1$ to $K$ **do**

  Compute Gradient: $\nabla g = X'(Y - X\beta_{i-1})/N$

  **repeat**

    $z = \beta_{i-1} - t \cdot \nabla g$

    $\beta_i = S_{\lambda,t}(z)$

    Compute: $G_t = \frac{1}{t}(\beta_{i-1} - \beta_i)$

    Compute:

$$\text{LHS} = g(\beta_i), \quad \text{RHS} = g(\beta_{i-1}) - t\nabla g^\top G_t + \frac{t}{2}\|G_t\|_2^2$$

    **if** LHS $\leq$ RHS **then**

      **break**

    **else**

      $t = \eta \cdot t$

    **end if**

  **until** condition is met

  Compute Error

  **if** Error $<$ Tol **then**

    **break**

  **end if**

**end for**

### 2.5.2 The Fast Iterative Shrinkage Thresholding Algorithm (FISTA)

One of the disadvantages of the ISTA algorithm is its slow convergence rate, especially when solving problems with large-scale datasets. To address this issue, the FISTA (Fast Iterative Shrinkage-Thresholding Algorithm) [3] was developed a few years later as an accelerated version of ISTA.

The FISTA algorithm presents a better convergence rate $O(1/k^2)$ than ISTA by introducing an additional momentum variable $z_i$ that updates through a momentum parameter $t_i$ in each iteration based on the previous gradient.
This process speeds up the algorithm's move to the optimal solution. As a result, FISTA is faster than ISTA's $O(1/k)$ convergence rate. Compared to other alternative algorithms like **Coordinate Descent** and **ADMM**(Alternating Direction Method of Multipliers), FISTA maintains the computational simplicity of ISTA while increasing its convergence speed, making it suitable for dealing with large and high-dimensional datasets.

**Algorithm 3** : FISTA (Fast Iterative Shrinkage-Thresholding Algorithm)

**Inputs:**
$Y$ , $X$, $\lambda$,
$K$
Tol
**Initialize:**
$\beta_0$ (initial coefficients)
$z_0 = \beta_0$ (momentum variable)
$L =$ Lipschitz constant
step size $= 1/L$
$t_{\text{old}} = 1$
**for** $i = 1$ to $K$ **do**
    Compute Gradient : $\nabla g = X'(Y - Xz_{i-1})/N$
    Update coefficients: $\beta_i = S_{\lambda,\text{step size}}(z_{i-1} - \text{step size} \times \nabla g)$
    Update: $t_i = \frac{1+\sqrt{1+4 \cdot t_{\text{old}}^2}}{2}$
    Update: $z_i = \beta_i + \frac{t_{\text{old}}-1}{t_i}(\beta_i - \beta_{i-1})$
    Compute Error
    **if** Error $<$ Tol **then**
        **break**
    **end if**
    Update $t_{\text{old}} \leftarrow t_i$
**end for**

Although FISTA is a faster variant of ISTA due to its acceleration step, incorporating an adaptive step size via backtracking line search can further enhance convergence speed and stability.

**Algorithm 4** FISTA with Backtracking Line Search

**Inputs:** $Y$, $X$, $\lambda$, $K$, Tol

**Initialize:**

  $\beta_0$ (initial coefficients)

  $z_0 = \beta_0$ (momentum variable)

  $L$ = initial Lipschitz estimate

  $t = 4/L, \quad \eta \in (0, 1)$ (e.g., 0.8)

  $t_{\text{old}} = 1$

**for** $i = 1$ to $K$ **do**

  Compute Gradient: $\nabla g = X'(Y - Xz_{i-1})/N$

  **repeat**

    $z = z_{i-1} - t \cdot \nabla g$

    $\beta_i = S_{\lambda,t}(z)$

    $G_t = \frac{1}{t}(z_{i-1} - \beta_i)$

    Compute:

$$\text{LHS} = g(\beta_i), \quad \text{RHS} = g(z_{i-1}) - t\nabla g^\top G_t + \frac{t}{2}\|G_t\|_2^2$$

    **if** LHS $\leq$ RHS **then**

      **break**

    **else**

      $t = \eta \cdot t$

    **end if**

  **until** condition is met

  Update: $t_i = \frac{1+\sqrt{1+4t_{\text{old}}^2}}{2}$

  Update: $z_i = \beta_i + \frac{t_{\text{old}}-1}{t_i}(\beta_i - \beta_{i-1})$

  Compute Error

  **if** Error $<$ Tol **then**

    **break**

  **end if**

  Update $t_{\text{old}} \leftarrow t_i$

**end for**

# Chapter 3

# The package: ProxReg

In the fields of biology and public health, data characterized by a large number of variables that often exceed the number of observations is commonly referred to as high-dimensional data. Researchers frequently need to analyze this type of data to make predictions or classifications. However, traditional methodologies, such as linear regression, are typically unsuitable for high-dimensional data due to issues like multicollinearity and overfitting. To address these challenges, various methods and models have been developed, and numerous R packages have been created to support these approaches.

`ProxReg` is specifically designed to handle high-dimensional data by implementing models such as Ridge regression and Lasso regression. Additionally, the package incorporates standard linear regression for situations when simpler datasets are involved, thereby ensuring consistency and efficiency by eliminating the need to switch between different packages. It also integrates cross-validation, making data manipulation and model evaluation more streamlined. Moreover, it includes tools for assessing model accuracy, such as cross-validation and functions for calculating and showing root mean squared error (RMSE).

## 3.1 Overview of Functions

The `ProxReg` package provides a set of functions for fitting and evaluating regression models in both low- and high-dimensional datasets. Its core functions implement Lasso regression models using proximal gradient-based algorithms, specifically ISTA and FISTA. Additionally, the package supports ordinary least squares (OLS) regression and Ridge regression for simpler cases.

The package's primary functions fall into three categories:

- **Model fitting functions**: These include : `lasso_fista()`, `lasso_fista_back()`, `lasso_ista()`, `lasso_ista_back()`, `lasso_multi()`, `lasso_multi_back()`, `ols()` and `ridge()` . Each function follows a standard input-output structure:

  - **Inputs**: A dataset, a vector with names of predictor variables, a name of a response variable, and other options.

  - **Outputs**: Estimated coefficients along with optional visualizations or tables illustrating convergence status over iterations or significance tests for coefficients.

- **Evaluation functions**: Functions like `k_fold_cross()`, `get_rendiment()` and `visualize_error()` allow users to assess or compare performance between models using cross-validation, error metrics, and graphics that compare coefficient differences.
- **Utility functions**: Additional helper functions are provided to support data preparation and algorithmic operations. Examples include `softmax()` for probabilities of multiclass variables, `delete_rect()` for creating rectangular holes in images, and `inpainting()` for image reconstruction using Lasso regression.

## 3.2 Package Development

Figure 3.1 illustrates the development pipeline for the `ProxReg` R package. This process includes designing and coding main functions, adding documentation and examples, validating the package with *R CMD check* and submitting the package to CRAN for review. Moreover, the development of the `ProxReg` package was carried out using the following tools:

**Development Environment**

- **RStudio** : the main workspace used to write, test, and organize the package code
- **devtools** and **usethis** : for project setup, dependency management, and structure automation
- **roxygen2** : for automatic documentation generation directly inside the functions
- **R CMD check** : for the package validation before submission

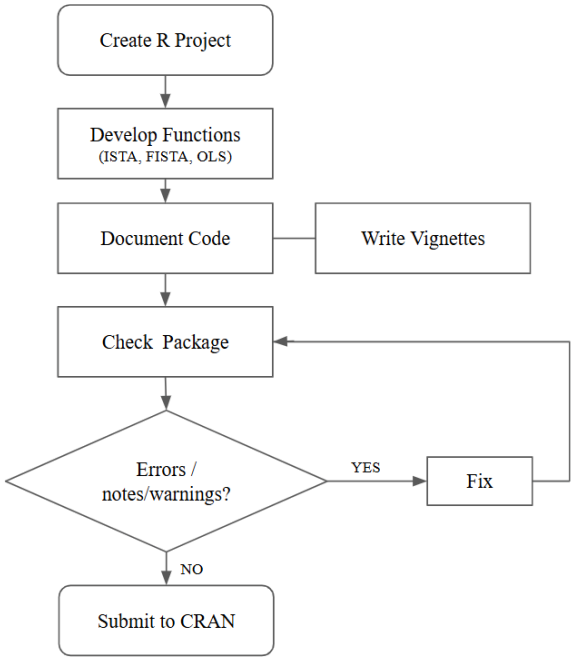


Figure 3.1: Workflow for the ProxReg package creation

The package was submitted and accepted on CRAN and published on March 15, 2025, and is now publicly available for download. An example of the usage of `ProxReg` is illustrated below:

```
install.packages("ProxReg")
library(ProxReg)
Example usage:
model <- lasso_fista(data = df , x = c("x1", "x2", x3),
y = "y", lambda = 0.1, max_step=10000,tol=0.001)
```

## 3.3 Function Explanations

### 3.3.1 Ordinary Least Squares Regression

The ordinary least squares regression is one of the most common uses of linear regression to estimate the relationship between one target variable and one or more independent variables. In *ProxReg* package, this method is implemented by using the *ols()* function, which returns the estimated coefficients of the linear regression model by minimizing the sum of squared error. Below is a brief overview of how the function works.

**Function syntax**

```
model <- ols(data,y,x,alpha=0.025,verbose=TRUE)
```

```
Parameters
```

- `data`: The dataset with the target variable and predictors
- `y`: The name of the target variable
- `x`: The name or names of the predictors
- `alpha`: The significance level for the coefficient estimators. The default value is 0.025, corresponding to a $95\%$ confidence level for two-sided intervals.
- `verbose`: A logical parameter. If it is TRUE (the default), the function will return a detailed table of the coefficients.

### 3.3.2 Ridge Regression

Ridge regression is a regularized linear regression that reduces model complexity and prevents overfitting by adding an L2 penalty to the loss function. The regularization strength is controlled by the lambda parameter, where larger values of lambda lead to more regularization. The *ridge()* function of the package implements this linear model.

```
Function syntax
```

```
model <- ridge(data,y,x,lambda)
```

```
Parameters
```

- `data`: The dataset with the target variable and predictors
- `y`: The name of the target variable
- `x`: The name or names of the predictors
- `lambda`: A number or numeric vector representing the penalty term

### 3.3.3 Lasso Regression

Lasso regression functions is similar to Ridge regression, instead of adding an L2 penalty to the loss function, they use an L1 penalty that can drive some coefficients to exactly zero and allows feature selection. The functions in the package are highly developed for Lasso regression and adapt to different types of target variables under the ISTA and FISTA algorithms, which are efficient methods for solving Lasso regression problems.

The `ProxReg` package implements Lasso regression using efficient gradient-based optimization algorithms: **ISTA** (Iterative Shrinkage-Thresholding Algorithm) and **FISTA** (Fast ISTA). These methods are particularly suitable for large-scale problems due to their speed and convergence properties, following are some examples.

**Function syntax**

```
model <- lasso_fista(data, y, x, max_step=10000,
tol=1e-7, lambda = 0.1, image=TRUE, type="Gaussian")

model <- lasso_ista(data, y, x, max_step=10000,
tol=1e-7, lambda = 0.1, image=TRUE, type="Gaussian")
```

`Parameters`

- `data`: The dataset containing the predictors and the target variable.
- `y`: The name of the target variable (as a string).
- `x`: A character vector with the names of the predictor variables.
- `lambda`: A non-negative value that controls the strength of the penalty. Higher values increase sparsity.
- `max_step`: Maximum iterations to run the algorithm.
- `tol`: Error tolerance.
- `type`: Response variable type; if it is *Binomial*, logistic Lasso regression is used.

# Chapter 4

# Evaluation Metrics

A wide range of evaluation metrics [22] is employed in this study to assess model performances and compare the results across different algorithms. For regression models, the following metrics are used: mean absolute error (MAE), root mean squared error (RMSE), mean absolute percentage error (MAPE), R-squared ($R^2$), and adjusted R-squared.

For classification models, evaluation is based primarily on the confusion matrix and ROC curve to provide information about classification accuracy, sensitivity, specificity, and comparison between true and false positive rates.

## 4.1 Mean Absolute Error (MAE)

The mean absolute error [17] measures the mean value of the absolute differences between predicted values ($\hat{y_i}$) and the actual observed target values ($y_i$), and is calculated by the formula:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y_i}|$$

MAE is one of the most commonly used evaluation metrics [28] due to its simplicity, easy interpretation, and robustness to outliers. It expresses the average prediction error in the same units as the target variable, making it directly understandable. A lower value of MAE indicates better predictive performance, a value closer to zero indicates high model accuracy.

## 4.2 Root Mean Squared Error (RMSE)

The root mean squared error [17] quantifies the average magnitude of the prediction error by taking the squared root of the mean of the squared differences between the predicted values and

the actual values. It is defined as:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2}$$

RMSE shares similar characteristics with MAE. It shares the same unit as the target variable and serves as a measure of how close the predicted values are to actual values. The value of RMSE ranges from 0 to infinity, where a value of 0 indicates perfect predictive accuracy. When RMSE values are lower, model performance is better.

## 4.3 Mean Absolute Percentage Error (MAPE)

The mean absolute percentage error [7] is another commonly used metric for evaluating the accuracy of the predictive models and measures the average absolute percentage difference between the predicted values and the actual values.

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y_i}}{y_i}\right| * 100$$

MAPE expresses prediction errors as a percentage of the actual values, which allows error comparison across datasets with different scales. A lower MAPE indicates higher model accuracy with $0\%$ as a perfect prediction. However, it presents some limitations and becomes unstable when the actual values are zero or near zero.

## 4.4 The Coefficient of Determination: R-squared and Adjusted R-squared

The $R^2$ score [6] is a measure of the goodness of fit of a model, it quantifies the proportion of variance in the dependent variable that is explained by the independent variables. The $R^2$ score ranges from 0 to 1, where higher values indicate a better fit. Mathematically, the $R^2$ is defined as:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y_i})^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

Where $\hat{y_i}$ is the predicted value, $y_i$ the actual value and $\bar{y}$, the mean of the observed values. One limitation of $R^2$ is that it always increases when more predictors are added to the model, therefore, it is usually complemented by adjusted $R^2$ which penalizes the addition of irrelevant variables and has the following mathematical expression:

$$Adjusted\ R^2 = 1 - \frac{(1 - R^2)(N - 1)}{N - p - 1}$$

Where *N* is the total sample size, and *p* is the number of independent variables.

## 4.5 Confusion Matrix

In classification tasks, model performance is evaluated using different metrics than those applied in regression problems. The most commonly used methods are the confusion matrix and the ROC curve.

A confusion matrix [27] is a numeric table that compares the actual class labels in rows with the predicted class labels presented in columns. Each cell in the matrix corresponds to one of the four possible terms:

- True Positives (TP): number of correct identification for the positive class
- True Negative (TN): number of correct identification for the negative class
- False Positive (FP): number of incorrect identification for the positive class
- False Negative (FN): number of incorrect identification for the negative class

In this study, the confusion matrix is constructed using the function *confusionMatrix()* from the `caret` package in R. In addition to presenting the counts of classification outcomes, it also returns a comprehensive set of evaluation statistics [33]. These include accuracy, sensitivity, specificity, false positive rate, and false negative rate, among other metrics, and are calculated with the following formulas.

- accuracy / true positive rate = $\frac{TP+TN}{TP+TN+FP+FN}$
- sensitivity = $\frac{TP}{TP+FN}$
- specificity = $\frac{TN}{TN+FP}$
- false positive rate = $\frac{FP}{FP+TN}$
- false negative rate = $\frac{FN}{FN+TP}$

## 4.6 ROC Curve

The ROC curve is a visual tool that illustrates the performance of classification models. It is constructed by plotting the true positive rate, equivalent to sensitivity, against the false positive rate (1 - specificity).

A model that achieves perfect classification will have a point located at the top-left corner of the plot, meaning high sensitivity and a low false positive rate. In contrast, a curve that lies close to the diagonal line suggests that the model performs no better than random guessing.

Often, the Area Under the Curve (AUC) [4] is calculated alongside the ROC curve as a numerical summary of a classification model's performance. It represents the area under the ROC curve and ranges from 0.5 to 1. An AUC of 1 indicates perfect classification ability, while an AUC of 0.5 suggests that the model has no significant classification ability [19].

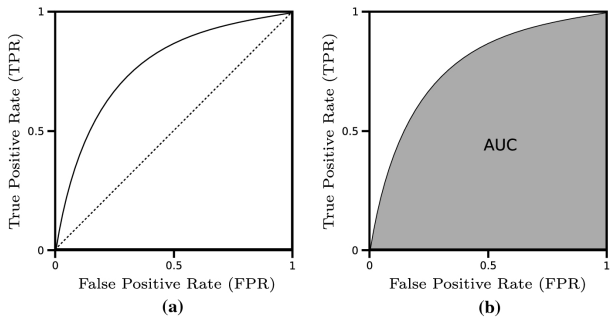


Figure 4.1: ROC curve and AUC

# Chapter 5

# Experiments

The results of applying the `ProxReg` package to both simulated and real-world datasets are presented in this section. The performance of the proximal gradient-based methods, specifically ISTA and FISTA, is compared with the coordinate descent algorithm implemented in the `glmnet` package. The evaluation focuses on key aspects such as the accuracy of coefficient estimation, convergence rate, and predictive performance, with results illustrated using tables and visualizations. Applications of the primary functions across various data types are discussed throughout the chapter.

## 5.1 Real-World Datasets

### 5.1.1 Metabric

Breast cancer is currently one of the most common cancers among women. Until 2022, over 2.3 million women worldwide have been diagnosed with breast cancer. The *Metabric* dataset collects clinical data and gene data from 1904 primary breast cancer samples and includes 693 variables, such as tumor size, mutation count, and survival months, among others. The dataset has missing values in only four columns, with tumor stage having the highest proportion of missing data ($26.31\%$). To handle these missing values, the Multiple Imputation by Chained Equations (MICE) technique was employed, which generates 5 imputed datasets by modeling the missing values as a function of other observed variables in the dataset.

For the model application, 190 mutation-related variables were eliminated from the original dataset, as they were binary and predominantly took the value of 0. These variables contributed minimal information for predicting the target variable. Moreover, removing these variables improved the model's performance by increasing computational efficiency and reducing unnecessary complexity, allowing the model to run faster.

### 5.1.2 Amazon

The Amazon dataset was obtained from *Yahoo Finance* and collected daily Amazon stock price data from March 16, 2009, to December 30, 2019.

The dataset does not present any missing values and consists of 2718 observations and 13 variables, including stock prices, trading volume, and several manually calculated financial indicators such as SMA (sample moving average), RSI (relative strength index), and EMA (exponential moving average), selected for their common application in financial modeling as indicators that reflect price trends, volatility, and momentum. Additionally, the previous day's price is added into the dataset as a lagged feature that captures temporal dependencies. The target variable for prediction in this dataset is the closing price of daily Amazon stock.

### 5.1.3 Glmnet Multinomial Dataset

The `glmnet` package not only provides algorithms for regularized regressions, but also includes a collection of example datasets that facilitate model demonstration and evaluation. One dataset is designed for multinomial classification and contains 500 observations, and 30 predictors, and a categorical target variable with three distinct classes.

Such a dataset is used to assess the ability of various Lasso-based methods to correctly classify multiclass responses and select relevant predictors.

### 5.1.4 Bird Image

Lasso regression can also be applied to image reconstruction tasks. The Bird dataset consists of pixel values from a colored bird image with a resolution of 256 x 256 pixels across 3 color channels representing the RGB colors: red, green, and blue. The data is organized into 3 matrices, each of size 256 x 256, corresponding to the individual color channels.

The pixel values in each matrix range from 0 to 1 and represent the intensity of each color in the image. Figure 5.1 illustrates the specific bird image used for this dataset, demonstrating detailed color intensities and patterns.

Figure 5.1: Bird

## 5.2 Simulated Datasets

One drawback of using real-world data is the absence of known coefficient values, which impedes a direct evaluation of the model's accuracy in coefficient estimation. Hence, to overcome this limitation, simulated datasets with predefined coefficients will be generated to compare estimated values with the true parameters.

The package `simdata` provides a set of functions that generate different types of datasets under controlled statistical settings, and it supports the simulation of linear, logistic, and survival data structures with predetermined coefficient vectors, correlation structures, and error terms by users.

In this study, a simulated dataset with 60 observations and 101 variables was generated. It consisted of 100 correlated predictors and one response variable. The predictors followed a multivariate normal distribution which allows precise control over the correlation structure through a predefined correlation matrix, where the first 30 predictors are highly associated with a Pearson correlation coefficient of 0.8, the next 20 predictors are weakly correlated with a coefficient of 0.2, and the remaining 50 are mutually independent. This correlation structure is visualized in Figure 5.2 .

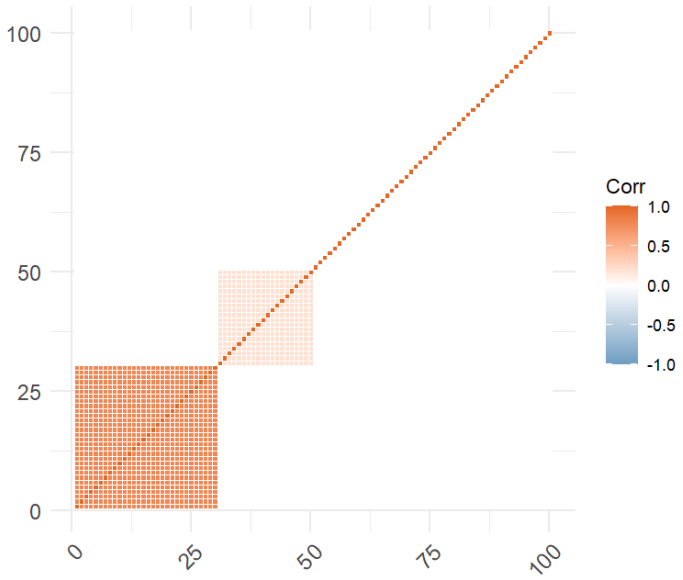


Figure 5.2: Heat map of correlation matrix

Finally, a continuous response variable is generated as a linear combination of the predictors, using a vector of predefined coefficients specified by the user, with added noise from a standard normal distribution.

## 5.3 Experimental Configuration

### 5.3.1 Data splitting

In model evaluation, it is essential to partition the dataset into a training set and a test set to avoid overfitting. Without this separation, the model may fit the training data too well and fail to generalize to unseen data and lead to poor predictive performance on new observations.

For each dataset used in this study, the data was randomly split into a training set containing 80% of the total observations and a test set comprising the remaining 20%. The remaining set was used to fit the models, while the test set served to evaluate their predictive performance.

### 5.3.2 Hyperparameter Setting

In model training, hyperparameter tuning is important to optimize the performance of a model. For the Lasso regression models implemented in the `ProxReg` package, the primary hyperparameter requiring configuration is the regularization parameter $\lambda$ which controls the strength of the penalty to the estimated coefficients. As the value of $\lambda$ increases, a greater number of coefficients become exactly zero.

In this study, hyperparameter selection was aligned with the type of target variable. For continuous response variables, the $\lambda$ value was chosen to minimize the mean squared error, while for categorical response variables, it was selected to maximize the classification accuracy.

## 5.4 Experiment Results

### 5.4.1 Metabric

Following the data splitting procedure, the Metabric dataset was partitioned into 80% for training and 20% for testing. The dataset used for model training contained 1523 observations and 502 variables, including both clinical and genomic information.

The dataset was used to carry out two predictive modeling tasks. The first involved predicting the number of survival months using Lasso regression, and the second task focused on classifying patient survival status, where the target variable was encoded as a binary indicator with 1 representing deceased and 0 representing survived. Both tasks implemented models with fixed step size and backtracking line search techniques under ISTA, FISTA, and coordinate descent algorithms.

The regularization parameter $\lambda$ was selected using cross-validation to mitigate overfitting, and

a vector of candidate values ranging from 0.002 to 1 with an increment of 0.005 was evaluated to identify the one that optimized the model's predictive performance. For the Lasso regression implemented in `ProxReg`, the maximum number of iterations was set to 10000 and a tolerance of $10^{-4}$ was defined.

The **table** 5.1 presents the performance metrics for each Lasso regression implementation in estimating survival months. Among all methods, the FISTA algorithm with backtracking line search achieved the lowest RMSE, RMAE and MAPE, along with a high value of $R^2$, which indicates a high predictive accuracy and minimal error.

| **Method** | **RMSE** | **RMAE** | **MAPE** | $\boldsymbol{R^2}$ |
|---|---|---|---|---|
| ISTA (fixed step) | 5.9037 | 2.1086 | 8.2364 | 0.9941 |
| FISTA (fixed step) | 0.0305 | 0.1548 | 0.0380 | 0.9999 |
| ISTA (line search) | 4.4169 | 1.8366 | 6.2156 | 0.9967 |
| FISTA (line search) | 0.0268 | 0.1437 | 0.0343 | 0.9999 |
| glmnet (coord. descent) | 2.2353 | 1.3613 | 3.5239 | 0.9991 |

Table 5.1: Performance metric of Lasso on Metabric dataset

In the classification task, the binary target variable is the survival status of breast cancer patients, where 1 indicates that the patient is alive and 0 indicates deceased. To predict the survival status based on genomic and clinical information, Lasso logistic regression models were implemented using the same optimization algorithms applied in the continuous prediction task, and model performance was assessed using confusion matrices and ROC curves.

Figure 4.1 compares the performance between ISTA, FISTA, and glmnet through ROC curves and the AUC index. It can be observed that ISTA's ROC curve traces the diagonal, and its AUC of 0.5 indicates the model has failed to separate classes correctly. In contrast, FISTA and glmnet show AUC values of 0.84 and 0.85 respectively, indicating good performance of the classifiers. Both methods produce ROC curves that climb quickly toward the upper-left corner to reach true-positive rates above 80% while keeping false-positive rates below 30%, meaning the algorithms correctly classify the true cases most of the time.

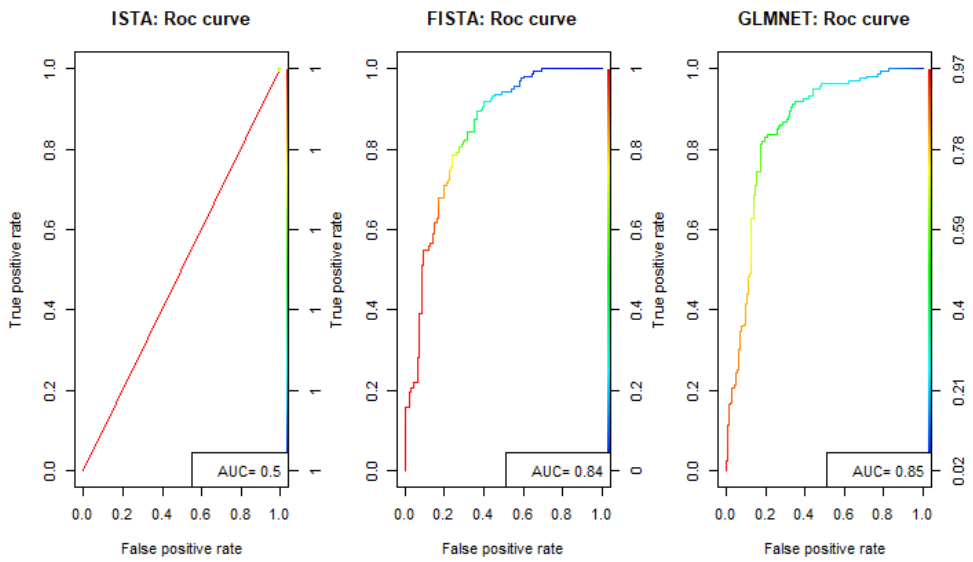


Figure 5.3: ROC curves for ISTA, FISTA, and glmnet

### 5.4.2 Amazon Stock Price

The target variable in the Amazon dataset was split chronologically into a training set and a test set. The training set contains 2174 stock price information points from 16 March 2009 to 30 October 2017, representing 80% of the whole dataset. And the remaining 20% of the test set covers the period from 31 October 2017 to 30 December 2019.

This splitting strategy ensures that the model is trained on past data and evaluated on future data. In this way, the temporal dependencies between consecutive observations are preserved, and the reliability of time series predictions is improved.

Figure 5.4 represents stock price historical data and predictions for Amazon with multiple colored lines that indicate different algorithm outcomes for Lasso regression models. The vertical line marks a specific point on prediction, and the model's performances can be compared by how close the predicted values are to the original stock prices. Visually, the ISTA algorithm with backtracking line search shows the closest prediction to the actual value for the specified date.


Figure 5.4: Amazon stock price predictions

Table 5.2 reflects the predictive performance of each algorithm on the dataset. Both models based on FISTA algorithms achieve the lowest values for RMSE, RMAE and MAPE, along with perfect $R^2$ scores indicating very high model explainability.

| **Method** | **RMSE** | **MAE** | **MAPE** | $R^2$ |
|---|---|---|---|---|
| ISTA (fixed step) | 1.4120 | 1.0427 | 1.3313 | 0.9841 |
| FISTA (fixed step) | 0.0108 | 0.0866 | 0.0092 | 1.0000 |
| ISTA (line search) | 0.9893 | 0.8729 | 0.9329 | 0.9922 |
| FISTA (line search) | 0.0106 | 0.0881 | 0.0094 | 1.0000 |
| glmnet (coord. descent) | 1.9139 | 1.3260 | 2.0943 | 0.9708 |

Table 5.2: Performance metrics of Lasso models on the Amazon dataset

### 5.4.3 Birds Image

The bird image used for the figure reconstruction task is composed of 3 square matrices of size 256 x 256 corresponding to the red, green, and blue color channels. The reconstruction process was carried out in three phases [25]. Firstly, a rectangular region of the picture was artificially corrupted by setting the pixel values in all three channels to -100 to move the original color information. This corrupted image is shown in Figure 5.5.

Figure 5.5: Figure with a noisy region

In the second phase, a set of small square regions, or patches, was extracted from various parts of the image to build a dataset of examples that contains both valid and damaged pixel values. And the pixel values surrounding the damaged regions served as target values to train a model able to estimate the missing information. Once trained on the dataset of undamaged pixel data, the model was applied to predict the pixel values in the corrupted region of the image.

In the final phase, the missing pixels were replaced with the predicted values, and the updated matrices with the reconstruction pixel information were used to generate and display the repaired version of the image.

Figure 5.6 presents results obtained using Lasso regressions through coordinate descent (`glmnet`), ISTA, and FISTA algorithms. The corrupted version of the original image is shown in the top-left corner, followed by the pictures repaired by using Lasso models. Visually, all methods restored the missing region with high precision. And from a computational perspective, the three methods required 24 iterations to complete the reconstruction task, however, the Lasso regression implemented via the `glmnet` package showed the highest efficiency, completing the task in the shortest time.

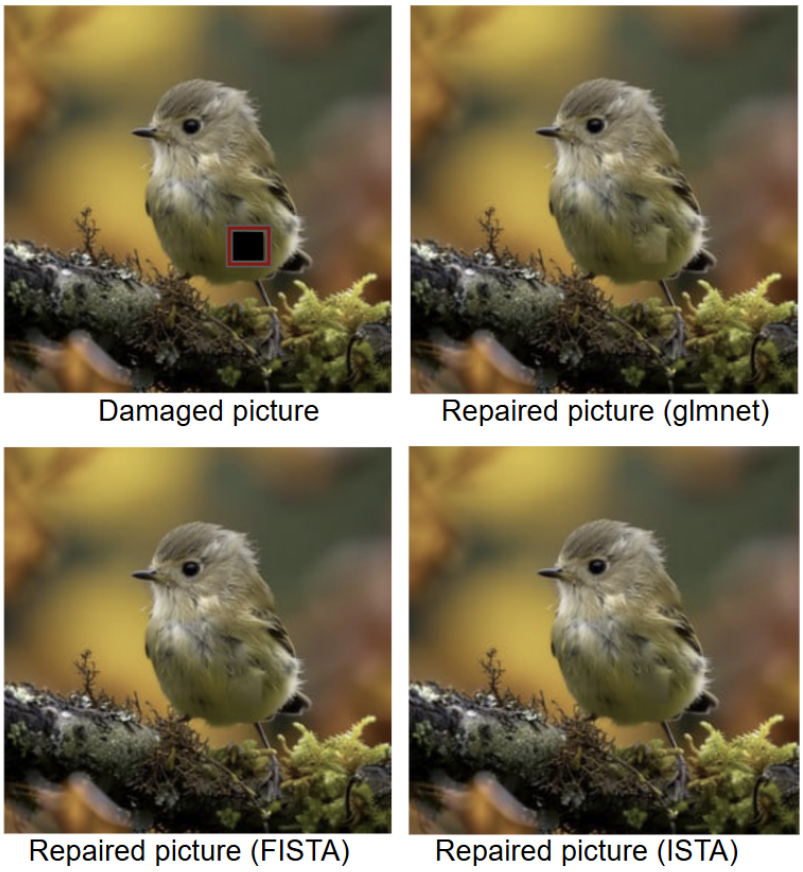


Figure 5.6: Repaired figure

### 5.4.4 Glmnet Multinomial Dataset

To evaluate the performance of multinomial logistic lasso regression models, the multinomial dataset provided by the *glmnet* package was used. The target variable is categorical with three imbalanced classes: class 1 (142 observations), class 2 (174 observations), and class 3 (184 observations). To prevent bias, the dataset was split into training and testing subsets using stratified sampling by class. In this way, the class proportions were preserved in both tests.

Figure 5.7 illustrates the classification performance of the FISTA, ISTA, and glmnet algorithms on the stratified test set. Each row corresponds to a true class, while the colored blocks indicate the predicted class, where green is for class 1, orange for class 2, and yellow for class 3. An ideal model with perfect predictive ability would produce homogeneous color blocks along each row. Overall, FISTA and glmnet achieve similar predictive accuracy across all three classes. ISTA also performs well, though with slight dispersion in prediction for classes 1 and 3.

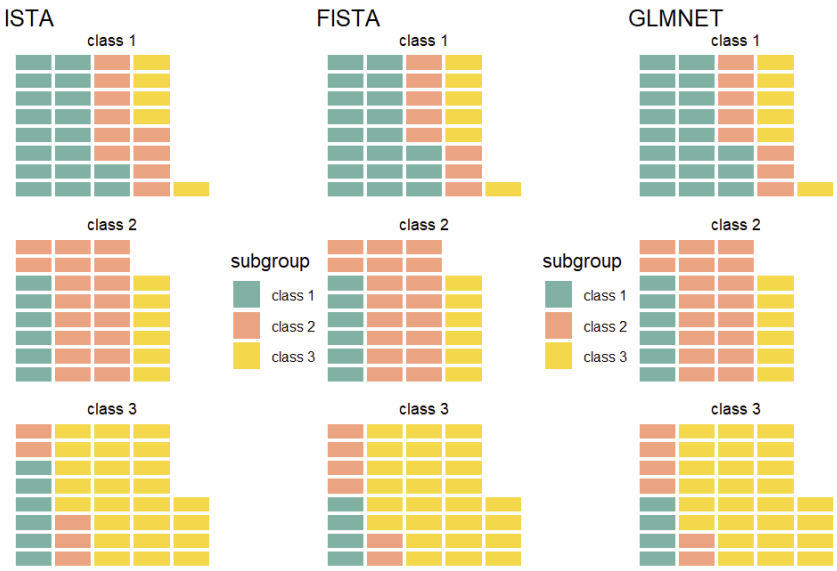


Figure 5.7: Multiclass performance

## 5.5 Simulated Dataset

The simulated dataset consists of 60 observations and 101 variables. The 100 predictors are denoted as $X_1, X_2, ..., X_{100}$ and the target variable ($Y$) is calculated using the following expression:

$$Y = 7.5 + 3.2X_1 + 1.5X_{10} - 2X_{25} + 1.8X_{50} - 0.5X_{75} + \epsilon$$

where $\epsilon$ represents a random error following a standard normal distribution $\epsilon \sim N(0, 1)$, with a mean of 0 and a standard deviation of 1.

In this experimental setup, the mentioned Lasso regression algorithms are applied to estimate the coefficients, which are then compared against the true underlying coefficients to evaluate the accuracy and effectiveness of each method.

### 5.5.1 Results

According to table 5.3, in general all the algorithms present similar performance, with `glmnet` standing out as the method achieving the lowest error across all metrics.

| **Method** | **RMSE** | **MAE** | **MAPE** | $\boldsymbol{R^2}$ |
|---|---|---|---|---|
| ISTA (fixed step) | 1.7063 | 1.2511 | 7.2514 | 0.9490 |
| FISTA (fixed step) | 1.1349 | 0.9788 | 3.9214 | 0.9775 |
| ISTA (line search) | 1.2740 | 1.0535 | 4.7323 | 0.9716 |
| FISTA (line search) | 1.1349 | 0.9787 | 3.9212 | 0.9775 |
| glmnet (coord. descent) | 1.0082 | 0.9337 | 4.2437 | 0.9822 |

Table 5.3: Performance metrics of Lasso models on simulated dataset

Figure 5.8 presents a comparison between estimated coefficients obtained by five algorithms with the true coefficients used to generate the data.

The majority of predictors in the dataset are irrelevant with a zero coefficient, and only five predictors are active. The figure uses color to encode the magnitude of estimation errors, while the gray color represents shared zero in both estimated and real coefficients.

All algorithms are able to identify most of the irrelevant features correctly, as seen by the abundance of grey blocks in the graphic. However, glmnet, which uses the coordinate descent algorithm, provides overall more accurate outcomes, especially for zero coefficients.

In the case of intercept estimation, as shown as the first red rectangles in the picture, all methods perform poorly in their estimation, producing an estimation error of more than 1.

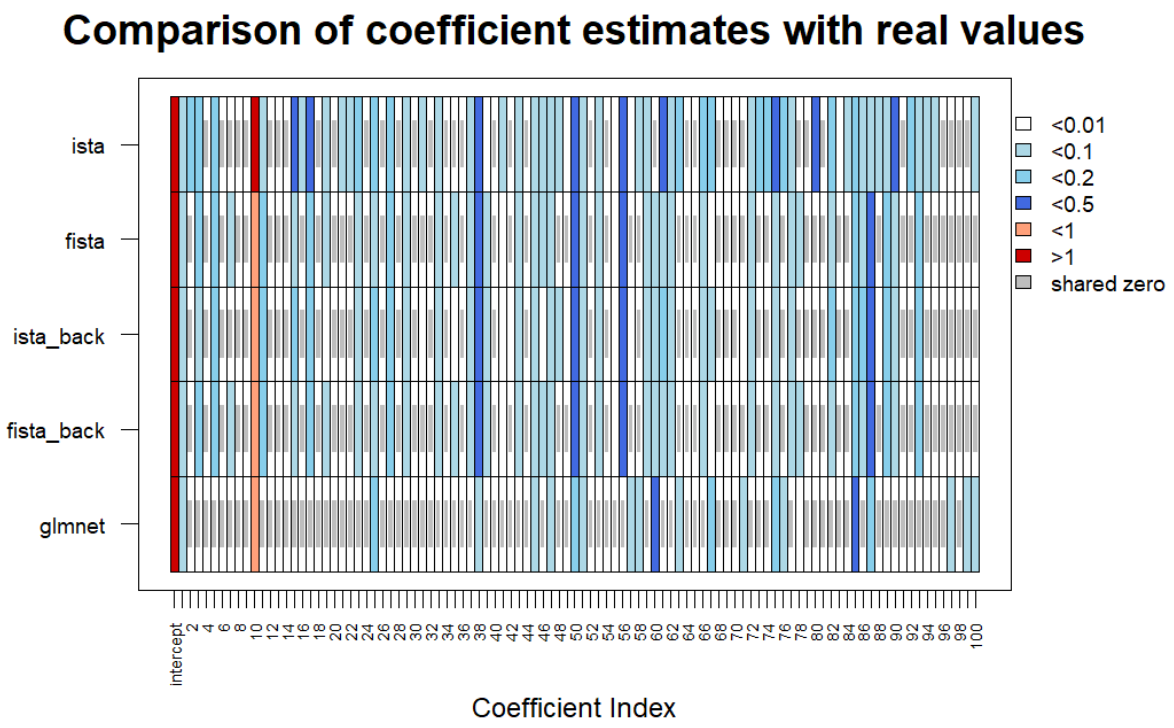


Figure 5.8: Comparison between coefficient estimates and real values

### 5.5.2 Feature Selection Consistency

In addition to the results on specific datasets, the study also aims to evaluate the behavior of Lasso-based algorithms implemented in both `ProxReg` and `glmnet` packages.

In terms of execution time, `glmnet` package employs a highly optimized coordinate descent algorithm written in Fortran, a low-level compiled language widely used in numeric computing due to its speed and efficiency. In contrast, the ISTA and FISTA algorithms were implemented manually in R using gradient-based update rules. Although both perform well theoretically, with FISTA having a faster convergence rate than basic coordinate descent, in practice, they are slower due to the reliance on iterative matrix operations and lack of compiled code like Fortran in glmnet [15].

Figure 5.9 shows the frequency of features over 50 random training sets generated from the simulated dataset using ISTA, FISTA and glmnet. Each line represents the number of times a feature was selected.

Among the three models, ISTA shows the highest frequency in general, which indicates it is less selective and tends to keep more variables. `glmnet`, by contrast, retains fewer variables, and FISTA falls between the two, showing a moderate selection frequency.

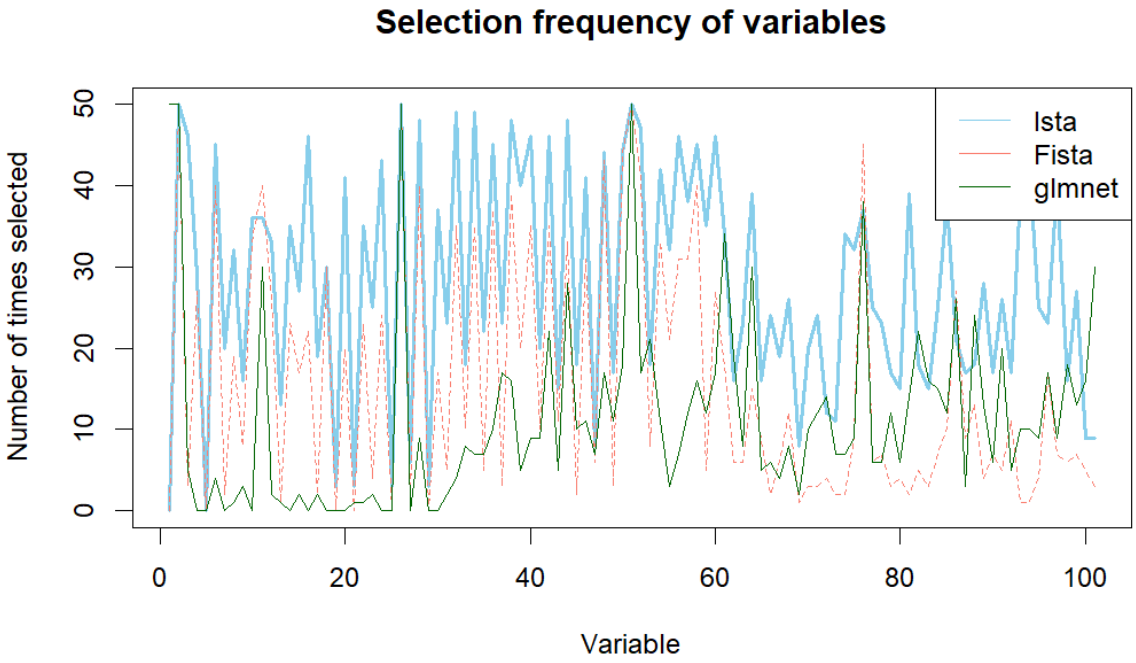


Figure 5.9: Feature consistency

Table 5.4 presents the ten most frequently selected predictors for ISTA, FISTA, `glmnet` across 50 independently simulated training sets. The true informative variables in the simulated model are : $X_1, X_{10}, X_{25}, X_{50}$ and $X_{75}$, and all three methods correctly ranked $X_1, X_{25}$ and $X_{50}$ at the top. FISTA and glmnet also include $X_{75}$ and $X_{10}$ as relevant variables, meaning that, in general, both methods achieve a higher detection rate of true predictors than ISTA under the same testing condition.

| Rank | ISTA | FISTA | GLMNET |
|---|---|---|---|
| 1 | X1 | X1 | X1 |
| 2 | X25 | X25 | X25 |
| 3 | X50 | X50 | X50 |
| 4 | X31 | X75 | X75 |
| 5 | X33 | X49 | X60 |
| 6 | X27 | X47 | X10 |
| 7 | X37 | X51 | X63 |
| 8 | X43 | X5 | X100 |
| 9 | X51 | X10 | X43 |
| 10 | X2 | X27 | X85 |

Table 5.4: Rankings of variables across ISTA, FISTA, and glmnet methods

Figure 5.10 shows the overlap among the top 10 predictors selected by the ISTA, FISTA, and GLMNET algorithms. Among the five true predictors ($X_1$, $X_{10}$, $X_{25}$, $X_{50}$, and $X_{75}$), three ($X_1$, $X_{25}$, $X_{50}$) are consistently selected by all three methods, forming the central overlap of the Venn diagram. FISTA and `glmnet` jointly identify all five true predictors, while ISTA misses 2. This indicates that FISTA and `glmnbet` show higher consistency in detecting the relevant features. The remaining variables shown in the diagram correspond to other highly ranked but non-true predictors included in the top-10 lists of each method.

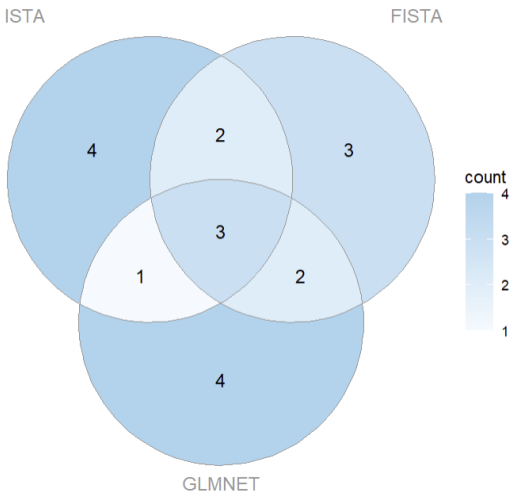


Figure 5.10: Venn Diagram for top-selected predictors

Table 5.5 summarizes the average predictive performance of each method. Among the three methods, FISTA achieves the best overall performance, with the lowest RMSE (1.6844), RMAE (1.1609), and MAPE (7.2355), which indicates the most accurate and consistent predictions on the test set. A $R^2$ value of 0.9751 suggests that FISTA explains a larger proportion of the variance in the response variable compared to the other algorithms. The glmnet method also performs well, with a strong $R^2$ of 0.9532 and moderate error metrics. Despite its relatively sparse and consistent feature selection, glmnet does not outperform FISTA. This effect may indicate that glmnet's stronger regularization and inclusion of fewer variables may limit its predictive accuracy. By contrast, ISTA shows the weakest performance, with the highest prediction error across all the metrics and a lower $R^2$, this may reflect its slower convergence and tendency to retain more irrelevant features.

| Method | RMSE | RMAE | MAPE | R squared |
|---|---|---|---|---|
| ISTA | 3.8183 | 1.7440 | 17.0258 | 0.8614 |
| FISTA | 1.6844 | 1.1609 | 7.2355 | 0.9751 |
| glmnet | 2.1546 | 1.2961 | 9.4109 | 0.9532 |

Table 5.5: Performance Metrics on simulated dataset

# Chapter 6

# Conclusion

The study focuses on developing and implementing proximal gradient-based algorithms, specifically ISTA and FISTA, for solving Lasso optimization problems in high-dimensional datasets. By integrating these methods into the `Proxreg` R package, it was possible to easily access the tools for model fitting and evaluation.

Through extensive experiments on both real-world and simulated datasets, a detailed comparison between the proximal-gradient based Lasso implementation and the coordinate descent algorithm provided by the widely-used `glmnet` package. While FISTA theoretically presents a faster convergence rate than ISTA and coordinate descent, in practice, `glmnet` method achieves a faster convergence due to its compiled Fortran code. Additionally, `glmnet` generally shows slightly better performance in coefficient estimation and more effective sparse feature selection in certain cases. However, FISTA proves to be a strong and competitive alternative due to its predictive performance and flexibility.

The `ProxReg` package offers a user-friendly and transparent solution for implementing Lasso regression. Its R-based implementation allows researchers to test and modify the methods without needing complicated external tools or special coding languages like Fortran. In contrast, `glmnet`, which is highly efficient, is less flexible for modification due to its reliance on low-level Fortran code.

# Chapter 7

# Reflections and Future Works

The development of the `ProxReg` package presented several challenges. One of the most significant aspects was solving practical issues such as numerical stability. For example, when implementing the softmax function, the denominator can be very large due to exponentials and can lead to numerical instability. To overcome this, adjustments were applied to the final equation to ensure numerical stability. This leads to the slight deviation from the original formulation in order to prevent overflow and other computational issues.

Developing functions for advanced optimization methods like FISTA and ISTA required not only a good understanding of the underlying mathematical principles but also careful attention to the nuances of R's syntax. Debugging and optimizing the code while maintaining algorithm efficiency and numerical stability was time-consuming.

Looking ahead, there are many aspects to be improved in both the package and the algorithms. Future work will focus on completing the vignettes for the package, providing users with detailed examples and documentation to facilitate the application of the package. Additionally, for the computational performance, I plan to explore the use of compiled languages such as Fortran, used in the `glmnet` package, to speed up the computational velocity of the algorithms making them more competitive with other highly optimized implementations implemented in glmnet.

However, I recognize that the algorithm still needs further improvement to achieve the efficiency and speed of methods in glmnet, particularly when dealing with very large datasets. With these works, it will ensure that `Proxreg` becomes a more robust and competitive tool than `glmnet`.

# Bibliography


[1] Hervé Abdi and Lynne J Williams. "Principal component analysis". In: *Wiley interdisciplinary reviews: computational statistics* 2.4 (2010), pp. 433–459.

[2] Charu C Aggarwal, Lagerstrom-Fife Aggarwal, and Lagerstrom-Fife. *Linear algebra and optimization for machine learning*. Vol. 156. Springer, 2020.

[3] Amir Beck and Marc Teboulle. "A fast iterative shrinkage-thresholding algorithm for linear inverse problems". In: *SIAM journal on imaging sciences* 2.1 (2009), pp. 183–202.

[4] Andrew P Bradley. "The use of the area under the ROC curve in the evaluation of machine learning algorithms". In: *Pattern recognition* 30.7 (1997), pp. 1145–1159.

[5] Alexander L Burton. "OLS (Linear) regression". In: *The encyclopedia of research methods in criminology and Criminal Justice* 2 (2021), pp. 509–514.

[6] Davide Chicco, Matthijs J Warrens, and Giuseppe Jurman. "The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation". In: *Peerj computer science* 7 (2021), e623.

[7] Arnaud de Myttenaere et al. "Mean Absolute Percentage Error for regression models". In: *Neurocomputing* 192 (2016). Advances in artificial neural networks, machine learning and computational intelligence, pp. 38–48. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2015.12.114. URL: https://www.sciencedirect.com/science/article/pii/S0925231216003325.

[8] John E Dennis Jr and Robert B Schnabel. *Numerical methods for unconstrained optimization and nonlinear equations*. SIAM, 1996.

[9] Ronald A Fisher. "The goodness of fit of regression formulae, and the distribution of regression coefficients". In: *Journal of the Royal Statistical Society* (1922), pp. 597–612.

[10] Jerome H Friedman, Trevor Hastie, and Rob Tibshirani. "Regularization paths for generalized linear models via coordinate descent". In: *Journal of statistical software* 33 (2010), pp. 1–22.

[11] Francis Galton. *Typical laws of heredity*. William Clowes and Sons, 1877.

[12] Francis Galton. “Regression towards mediocrity in hereditary stature”. In: *The Journal of the Anthropological Institute of Great Britain and Ireland* 15 (1886), pp. 246–263.

[13] Paul Geladi and Bruce R Kowalski. “Partial least-squares regression: a tutorial”. In: *Analytica chimica acta* 185 (1986), pp. 1–17.

[14] Karol Gregor and Yann LeCun. “Learning fast approximations of sparse coding”. In: *Proceedings of the 27th international conference on international conference on machine learning*. 2010, pp. 399–406.

[15] Trevor Hastie and Junyang Qian. “Glmnet vignette”. In: *Retrieved June* 9.2016 (2014), pp. 1–30.

[16] Trevor Hastie et al. *The elements of statistical learning: data mining, inference, and prediction*. Vol. 2. Springer, 2009.

[17] Timothy O Hodson. “Root mean square error (RMSE) or mean absolute error (MAE): When to use them or not”. In: *Geoscientific Model Development Discussions* 2022 (2022), pp. 1–10.

[18] Arthur E Hoerl and Robert W Kennard. “Ridge regression: Biased estimation for nonorthogonal problems”. In: *Technometrics* 12.1 (1970), pp. 55–67.

[19] Pablo A Jaskowiak, Ivan G Costa, and Ricardo JGB Campello. “The area under the ROC curve as a measure of clustering quality”. In: *Data Mining and Knowledge Discovery* 36.3 (2022), pp. 1219–1245.

[20] Jong Hae Kim. “Multicollinearity and misleading statistical results”. In: *Korean journal of anesthesiology* 72.6 (2019), pp. 558–569.

[21] Gary C McDonald. “Ridge regression”. In: *Wiley Interdisciplinary Reviews: Computational Statistics* 1.1 (2009), pp. 93–100.

[22] Gireen Naidu, Tranos Zuva, and Elias Mmbongeni Sibanda. “A review of evaluation metrics in machine learning algorithms”. In: *Computer science on-line conference*. Springer. 2023, pp. 15–25.

[23] Karl Pearson. “VII. Mathematical contributions to the theory of evolution.—III. Regression, heredity, and panmixia”. In: *Philosophical Transactions of the Royal Society of London. Series A, containing papers of a mathematical or physical character* 187 (1896), pp. 253–318.

[24] Mark Schmidt, Glenn Fung, and Rómer Rosales. “Fast optimization methods for l1 regularization: A comparative study and two new approaches”. In: *European conference on machine learning*. Springer. 2007, pp. 286–297.

[25] Bin Shen et al. “Image inpainting via sparse representation”. In: *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE. 2009, pp. 697–700.

[26] Robert Tibshirani. “Regression shrinkage and selection via the lasso”. In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 58.1 (1996), pp. 267–288.

[27] Kai Ming Ting. “Confusion Matrix”. In: *Encyclopedia of Machine Learning*. Ed. by Claude Sammut and Geoffrey I. Webb. Boston, MA: Springer US, 2010, pp. 209–209. ISBN: 978-0-387-30164-8. DOI: 10.1007/978-0-387-30164-8_157. URL: https://doi.org/10.1007/978-0-387-30164-8_157.

[28] Cort J Willmott and Kenji Matsuura. “Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance”. In: *Climate research* 30.1 (2005), pp. 79–82.

[29] Stephen J Wright. “Coordinate descent algorithms”. In: *Mathematical programming* 151.1 (2015), pp. 3–34.

[30] Mengyuan Zhang and Kai Liu. “On regularized sparse logistic regression”. In: *2023 IEEE International Conference on Data Mining (ICDM)*. IEEE. 2023, pp. 1535–1540.

[31] Zhongheng Zhang. “Variable selection with stepwise and best subset approaches”. In: *Annals of translational medicine* 4.7 (2016), p. 136.

[32] Yujie Zhao and Xiaoming Huo. “A survey of numerical algorithms that can solve the Lasso problems”. In: *Wiley Interdisciplinary Reviews: Computational Statistics* 15.4 (2023), e1602.

[33] Wen Zhu, Nancy Zeng, Ning Wang, et al. “Sensitivity, specificity, accuracy, associated confidence interval and ROC analysis with practical SAS implementations”. In: *NESUG proceedings: health care and life sciences, Baltimore, Maryland* 19 (2010), p. 67.

# Appendix

## 7.1 Appendix A: Code Repository

The `ProxReg` package source code is available at:

github

## 7.2 Appendix B: CRAN Submission Details

**ProxReg: Linear Models for Prediction and Classification using Proximal Operators**

Implements optimization techniques for Lasso regression, R.Tibshirani(1996)<doi:10.1111/j.2517-6161.1996.tb02080.x> using Fast Iterative Shrinkage-Thresholding Algorithm (FISTA) and Iterative Shrinkage-Thresholding Algorithm (ISTA) based on proximal operators, A.Beck(2009)<doi:10.1137/080716542>. The package is useful for high-dimensional regression problems and includes cross-validation procedures to select optimal penalty parameters.

| | |
|---|---|
| Version: | 1.1.2 |
| Imports: | dplyr, EBImage, glmnet |
| Suggests: | knitr, rmarkdown |
| Published: | 2025-06-17 |
| Author: | YingHong Chen [aut, cre], Ferran Reverter Comes [aut], Esteban Vegas Lozano [aut] |
| Maintainer: | YingHong Chen <yinghongchen1402 at gmail.com> |
| License: | MIT + file LICENSE |
| NeedsCompilation: | no |
| Language: | en-US |
| CRAN checks: | ProxReg results |

**Documentation:**

| | |
|---|---|
| Reference manual: | ProxReg.pdf |
| Vignettes: | Manual for the package: ProxReg (source, R code) |

**Downloads:**

| | |
|---|---|
| Package source: | ProxReg_1.1.2.tar.gz |
| Windows binaries: | r-devel: ProxReg_1.1.1.zip, r-release: ProxReg_1.1.1.zip, r-oldrel: ProxReg_1.1.1.zip |
| macOS binaries: | r-release (arm64): ProxReg_1.1.1.tgz, r-oldrel (arm64): ProxReg_1.1.1.tgz, r-release (x86_64): ProxReg_0.1.1.tgz, r-oldrel (x86_64): ProxReg_0.1.1.tgz |
| Old sources: | ProxReg archive |

**Linking:**

Please use the canonical form https://CRAN.R-project.org/package=ProxReg to link to this page.

Figure 7.1: Package details

## 7.3 Appendix C: Certificate of Presentation of Poster


Figure 7.2: Certificate of poster presentation to GRBIO workshop