

spareg: Sparse Projected Averaged Regression in R

Laura Vana-Gür 
TU Wien

Roman Parzer 
TU Wien

Peter Filzmoser 
TU Wien

Abstract

The **spareg** package for R builds ensembles of predictive generalized linear models for high-dimensional data. It employs an algorithm that combines variable screening and random projection techniques in each model of the ensemble to address the computational challenges of large predictor sets. By implementing this modeling approach in an accessible framework, **spareg** enables users to apply methods that have shown competitive predictive performance against state-of-the-art alternatives, while at the same time keeping computational costs low.

Designed with extensibility in mind, **spareg** implements the screening and random projection techniques, as well as the generalized linear models employed in the ensemble as S3 classes with user-friendly constructor functions. This modular design allows users to seamlessly integrate and develop new procedures, making the package a versatile tool for high-dimensional applications.

Keywords: random projection, variable screening, ensemble learning, R.

1. Introduction

The **spareg** package for R (R Core Team 2025) introduces a computationally efficient framework for high-dimensional generalized linear modeling, uniquely combining variable screening, random projections, and ensemble learning to handle settings where the number of predictors p far exceeds the sample size n i.e., $p > n$ or even $p \gg n$.

More specifically, **spareg** builds an ensemble of generalized linear models (GLMs) where, in each model of the ensemble, i) variables are first screened based on a measure of the utility of each predictor for the response, ii) the selected variables are then projected to a lower dimensional space (smaller than p) using a random projection matrix, iii) a GLM is estimated using the projected predictors. Additional sparsity in the coefficients of the original variables can be introduced through a thresholding parameter, which, together with the number of models in the ensemble, can be chosen using a validation set or via cross-validation. Finally, predictions are obtained by averaging over the marginal models in the ensemble.

The motivation of such an algorithm, which performs what we call a *sparse projected averaged regression* (SPAR) for both discrete and continuous data in the GLM framework, lies in its computational efficiency. Random projection is a computationally efficient method which linearly maps a set of points in high dimensions into a much lower-dimensional space, and random projection matrices have been traditionally generated from suitable distributions in a data-agnostic fashion. By projecting the original predictors to a dimension lower than p , estimation of the models on the reduced predictors can be done using standard methods.

However, random projection can suffer from noise accumulation for very large p , as too many irrelevant predictors are being considered for prediction purposes (Mukhopadhyay and Dunson 2020). Therefore, the screening step is advisable in order to eliminate the influence of irrelevant variables before performing the random projection, while also reducing computational costs. On the other hand, the ensemble approach mitigates the variability introduced by random sampling by averaging the results from multiple iterations (Thanei, Heinze, and Meinshausen 2017).

Different variants of the algorithm have been shown to perform well in terms of prediction power on a variety of data sets. For example, Mukhopadhyay and Dunson (2020) employ the algorithm with the data-agnostic, sparse random projection of Achlioptas (2003) combined with probabilistic marginal correlation screening. Furthermore, Parzer, Filzmoser, and Vana-Gür (2024) show that, when paired with a carefully constructed data-driven random projection, the algorithm performs superiorly in terms of predictions and variable ranking in settings with different degrees of sparsity in the coefficients and with correlated predictors.

Package **spareg** (Vana-Gür, Parzer, and Filzmoser 2025) implements this flexible framework, offering a diverse selection of screening coefficients, random projection matrices, and (un)penalized marginal GLMs, and is available from the Comprehensive R Archive Network (CRAN) at <https://CRAN.R-project.org/package=spareg>. A key feature of the software is its extensibility: users can tailor the choice of screening, projection, and modeling techniques to their data’s specific characteristics. For instance, the data-driven random projection proposed by Parzer *et al.* (2024) excels in settings with correlated predictors and unknown sparsity levels, and it preserves information about the original coefficients, enhancing interpretability. In contrast, data-agnostic random projections (e.g., Achlioptas 2003) may lose structural information, making them less suitable when interpretability is a priority. Similarly, certain screening methods, such as marginal correlation screening, perform well when predictors are weakly correlated, while others, like ridge-based screening, are more robust in the presence of multicollinearity (more details are provided in Section 2). The package provides user-friendly constructor functions for S3 classes, enabling users to integrate custom screening, projection, or marginal GLM procedures into the ensemble framework, as well as methods such as `plot()`, `predict()`, `coef()`, `print()` which allow users to more easily interact with the model output and analyze the results.

In R, several other packages focus on building ensembles where the dimensionality of the predictors is reduced. Most notably, package **RPEnsemble** (Cannings and Samworth 2021) implements the procedure in Cannings and Samworth (2017), where “carefully selected” random projections are used for projecting the predictors before they are employed in a classifier such as k nearest neighbor, linear or quadratic discriminant analysis. On the other hand, package **RaSEn** (Tian and Feng 2021) implements an algorithm for ensemble classification and regression problems, where random subsets of predictors are generated and the optimal one is chosen to train a weak learner on the basis of some criterion. Similarly, package **randomGLM** (Song and Langfelder 2022) implements an ensemble predictor based on bootstrap aggregation (bagging) of GLMs whose covariates are first randomly selected and then further reduced using forward regression according to the Akaike information criterion (AIC). For Python (Van Rossum *et al.* 2011), a similar workflow could be set up for building ensembles using the package **pycobra** (Guedj and Desikan 2018) by integrating **scikit-learn** models (Pedregosa, Varoquaux, Gramfort, Michel, Thirion, Grisel, Blondel, Prettenhofer, Weiss, Dubourg *et al.* 2011). Random projection tools are implemented in

sklearn.random_projection while GLM estimation on the projected predictors (both classical and penalized) can be done using **sklearn.linear_model** module. Alternatively, ensembles can be built by using native stacking. Since the recent integration of the H2O machine learning models (H2O.ai 2016) in Stata (StataCorp 2025), it is also possible to build ensembles of machine learning models (including GLMs) via stacking, boosting or bagging, but random subspace methods or similar approaches to dimensionality reduction in GLMs are not implemented at the time of writing.

Unlike existing tools, **spareg** integrates screening, random projection and ensemble learning into a cohesive workflow, enabling estimation and interpretation of GLMs in challenging high-dimensional scenarios. Its modular design allows users to seamlessly extend the package with custom methods. Furthermore, **spareg** is the first package to implement this specific algorithm for GLMs, providing a comprehensive toolbox for model fitting, prediction, and visualization. Through the implemented data-driven random projection which ensures interpretability of variable effects, the package is particularly advantageous for practical applications in fields such as genomics, imaging, and finance. By offering this integrated functionality, the package fills a critical gap in the current software landscape, where no other tool provides the same capabilities for GLMs in high-dimensional settings.

The rest of the paper is organized as follows: Section 2 provides an overview of the methods and the methodological details of the implemented algorithm. The package is described in Section 3 while Section 4 exemplifies how a new screening procedure, a new random projection and a new marginal model can be integrated in the package. Section 5 concludes.

2. Methods

Throughout the section we assume to observe high-dimensional data $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, where $\mathbf{x}_i \in \mathbb{R}^p$ is a predictor vector and $y_i \in \mathbb{R}$ is the response, with $p \gg n$. The predictor vectors are collected in the rows of the predictor matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$.

2.1. Variable screening

In high-dimensional modeling, the goal of variable screening is to reduce the predictor set by selecting a small subset of variables with a strong *utility* to the response variable. This initial selection enables more efficient downstream analyses by discarding less relevant predictors early in the modeling process, thus reducing computational costs and potential noise accumulation stemming from irrelevant variables (see e.g., Mukhopadhyay and Dunson 2020).

The field of variable screening is an active area of research, with numerous methods developed to address different data settings. While a comprehensive review is beyond the scope of this paper, this section provides a concise and selective overview of key approaches for variable screening in high-dimensional settings. A classic approach is sure independence screening (SIS), proposed by Fan and Lv (2007). This method uses the vector of marginal empirical correlations $\hat{\boldsymbol{\omega}} = (\hat{\omega}_1, \dots, \hat{\omega}_p)^\top \in \mathbb{R}^p$, where $\hat{\omega}_j = \text{Cor}(\mathbf{X}_j, \mathbf{y})$. Here, $\mathbf{y}$ is the $(n \times 1)$ vector of responses and $\mathbf{X}_j$ is the j -th column of the matrix of predictors. SIS screens predictors in a linear regression setting by selecting the variable set $\mathcal{A}_\delta$, which contains all variables for which $|\hat{\omega}_j| > \delta$, with δ being a suitable threshold. Under certain technical conditions, this screening coefficient vector has the *sure screening property*, i.e., the set of truly active variables is included in $\mathcal{A}_{\delta_n}$ with probability converging to one as $n \rightarrow \infty$. Extensions to SIS

include modifications for GLMs (Fan and Song 2010), where screening is performed based on the log-likelihood $\ell(\cdot)$ or the slope coefficient of the GLM containing only $\mathbf{X}_j$ as a predictor: $\hat{\omega}_j := \operatorname{argmin}_{\beta_j \in \mathbb{R}} \min_{\beta_0 \in \mathbb{R}} \sum_{i=1}^n -\ell(\beta_j, \beta_0; y_i, x_{ij})$, where x_{ij} is the j -th entry of the vector $\mathbf{x}_i$. However, both mentioned approaches face limitations related to the required technical conditions which can rule out practically possible scenarios where an important variable is marginally uncorrelated to the response due to their multicollinearity. To tackle these issues, Fan, Samworth, and Wu (2009) propose to use an iterative procedure where SIS is applied subsequently on the residuals of the model estimated in a previous step. Additionally, in a linear regression setting, Cho and Fryzlewicz (2012) propose using the tilted correlation, i.e., the correlation of a tilted version of $\mathbf{X}_j$ with $\mathbf{y}$ where the effect of other variables is reduced. Wang and Leng (2016) propose to take into account the correlation among the predictors by employing the high-dimensional ordinary least squares projection (HOLP) $\hat{\mathbf{w}} = \mathbf{X}^\top (\mathbf{X} \mathbf{X}^\top)^{-1} \mathbf{y}$, which is a ridge estimator where the penalty term converges to zero. For GLMs, joint feature screening via a sparsity-restricted maximum likelihood estimator—under an ℓ_0 norm—has been proposed in Xu and Chen (2014).

In order to tackle potential model misspecification, a rich stream of literature focuses on developing semi- or non-parametric alternatives to SIS. For linear regression, approaches include using the ranked correlation (Zhu, Li, Li, and Zhu 2011), (conditional) distance correlation (Li, Zhong, and Zhu 2012; Wang, Pan, Hu, Tian, and Zhang 2015) or quantile correlation (Ma and Zhang 2016). For GLMs, Fan, Feng, and Song (2011) extend Fan and Song (2010) by fitting a generalized additive model with B-splines. Further extensions for discrete (or categorical) outcomes include the fused Kolmogorov filter (Mai and Zou 2013), the mean conditional variance, i.e., the expectation in $\mathbf{X}_j$ of the variance in the response of the empirical distribution function of each predictor conditional on the response (Cui, Li, and Zhong 2015). Ke (2023) proposes a model-free method where the contribution of each individual predictor is quantified marginally and conditionally in the presence of the control variables as well as the other candidates by reproducing-kernel-based R^2 and partial R^2 statistics.

The R landscape for variable screening techniques is very rich. An overview of some notable packages on the Comprehensive R Archive Network (CRAN) includes the following packages. Package **SIS** (Saldana and Feng 2018), which implements the (iterative) sure independence screening procedure and its extensions, as detailed in Fan and Lv (2007), Fan and Song (2010), Fan, Feng, and Wu (2010). This package also provides functionality for estimating a penalized generalized linear model or a cox regression model for the variables selected by the screening procedure. Package **VariableScreening** (Li, Huang, and Dziak 2022) offers screening methods for independent and identically distributed (iid) data, varying-coefficient models, and longitudinal data, and includes techniques such as sure independent ranking and screening (SIRS), which ranks the predictors by their correlation with the rank-ordered response, or distance correlation sure independence screening (DC-SIS), a non-parametric extension of the correlation coefficient. Package **MFSIS** (Cheng, Wang, Zhu, Zhong, and Zhou 2024) provides a collection of model-free screening techniques including SIRS, DC-SIS, the fused Kolmogorov filter (Mai and Zou 2015), the projection correlation method using knock-off features (Liu, Liu, Ke, and Li 2022), among others. Additional packages implement specific procedures but their review is beyond the scope of the current paper.

Package **spareg** allows the integration of such (advanced) screening techniques using a flexible framework, which in turn enables users to apply various screening methods tailored to their

data characteristics in the algorithm generating the ensemble. This flexibility allows users to evaluate different strategies, ensuring that the most effective approach is chosen for the specific application at hand. Moreover, it incorporates probabilistic screening strategies, which can be particularly useful in ensembles, as they enhance the diversity of predictors across ensemble models. Instead of relying on a fixed threshold δ , predictors are sampled with probabilities proportional to their screening coefficients (see Mukhopadhyay and Dunson 2020; Parzer *et al.* 2024). Note that this is different from the random subspace sampling employed in, e.g., random forests.

2.2. Random projection tools

Package **spareg** has been designed to allow the incorporation of various random projection techniques, enabling users to tailor the procedure to their specific data needs. Below, we provide background information on different random projection techniques and an overview of existing software implementing random projections.

The random projection method relies on the Johnson-Lindenstrauss (JL) lemma (Johnson and Lindenstrauss 1984), which asserts that for each set of points in p -dimensional Euclidean space collected in the rows of $X \in \mathbb{R}^{n \times p}$ there exists a linear map $\Phi \in \mathbb{R}^{m \times p}$ such that all pairwise distances are approximately preserved within a factor of $(1 \pm \epsilon)$ for $m \geq m_l = \mathcal{O}(\epsilon^{-2} \log(n))$. Computationally, an attractive feature of the method for high-dimensional settings is that the bound does not depend on p .

The goal is to choose a random map Φ that satisfies the JL lemma with high probability given that it fulfills certain technical conditions. The literature focuses on constructing such matrices either by sampling them from some “appropriate” distribution, by inducing sparsity in the matrix and/or by employing specific fast constructs which lead to efficient matrix-vector multiplications. It turns out that the conditions are generally satisfied by nearly all sub-Gaussian distributions (Matoušek 2008). A common choice is the standard normal distribution $\Phi_{ij} \stackrel{iid}{\sim} N(0, 1)$ (Frankl and Maehara 1988) or a sparser version where $\Phi_{ij} \stackrel{iid}{\sim} N(0, 1/\sqrt{\psi})$ with probability ψ and 0 otherwise (Matoušek 2008). Another computationally simpler option is the Rademacher distribution where $\Phi_{ij} = \pm 1/\sqrt{\psi}$ with probability $\psi/2$ and zero otherwise for $0 < \psi \leq 1$, where Achlioptas (2003) shows results for $\psi = 1$ and $\psi = 1/3$ while Li, Hastie, and Church (2006) recommend using $\psi = 1/\sqrt{p}$ to obtain very sparse matrices. Further approaches include using the Haar measure to generate random orthogonal matrices (Cannings and Samworth 2017) or a non-sub-Gaussian distribution like the standard Cauchy, proposed by Li *et al.* (2006), for preserving approximate ℓ_1 distances in settings where the data is high-dimensional, non-sparse, and heavy-tailed. Structured matrices, which allow for more efficient multiplication, have also been proposed (see e.g., Ailon and Chazelle 2009; Clarkson and Woodruff 2013).

The conventional random projections mentioned above are data-agnostic. However, recent work has proposed incorporating information from the data either to select the “best” data-agnostic random projection or to directly inform the random projection procedure. Cannings and Samworth (2017) rely on the former approach and build an ensemble classifier where the random projection matrix is chosen by selecting the one that minimizes the test error of the classification problem among a set of data-agnostic random projections. Another data-driven approach to random projection for regression has been proposed by Ryder, Karnin, and Liberty (2019), who introduce a data-informed random projection using an asymmetric

transformation of the predictor matrix without using information of the response.

On the other hand, [Parzer et al. \(2024\)](#) propose to use a random projection matrix for GLMs which directly incorporates information about the relationship between the predictors and the response, rather than a projection matrix which satisfies the JL lemma. In the linear regression case [Parzer, Filzmoser, and Vana-Gür \(2025\)](#) provide a theoretical bound on the expected reduction in prediction error when using a projection which incorporates information about the true regression coefficients compared to a conventional random projection. In particular, they rely on the sparse embedding (CW) matrix of [Clarkson and Woodruff \(2013\)](#). The CW random projection matrix Φ is constructed as

$$\Phi = BD \in \mathbb{R}^{m \times p}, \quad (1)$$

where B is a $(p \times p)$ binary matrix, where for each column j an index is uniformly sampled from $\{1, \dots, m\}$ and the corresponding entry is set to one, and D is a $(p \times p)$ diagonal matrix, with entries $d_j \sim \text{Unif}(\{-1, 1\})$. [Parzer et al. \(2025\)](#) provide theoretical results when replacing the diagonal entries of D with the true regression coefficients. Motivated by this result, they propose to construct such a projection matrix where the random diagonal elements are replaced in practice by the HOLP estimates, i.e., the ridge coefficients with a penalty converging to zero. The constructed random projection has the advantage of approximately capturing the true regression coefficients in the span of the random projection, i.e., it ensures that the true regression coefficients can be recovered approximately after the projection. In [Parzer et al. \(2024\)](#), the authors extend this idea to the GLM framework by investigating how similar ridge coefficients can be used to construct a suitable data-informed projection matrix. Through simulations, they demonstrate that for non-Gaussian families, allowing the penalty to converge to zero is not always optimal. Instead, they propose selecting the smallest penalty value such that the deviance ratio (the proportion of the null deviance explained) remains below a specified threshold. Specifically, they suggest using a threshold of 0.8 for non-Gaussian families and 0.99 for Gaussian families.

Several packages in R provide functionality for random projections. For instance, package **RandPro** ([Aghila and Siddharth 2020](#); [Siddharth and Aghila 2020](#)) allows a Gaussian random matrix (i.e., where entries are simulated from a standard normal), a sparse matrix ([Achlioptas 2003](#); [Li et al. 2006](#)) or a matrix generated using the equal probability distribution with the elements $\{-1, 1\}$, to be applied to the predictor matrix before employing one of k nearest neighbors, support vector machine or naive Bayes classifier on the projected features. Package **SPCAvRP** ([Gataric, Wang, and Samworth 2019](#)) implements sparse principal component analysis, based on the aggregation of eigenvector information from “carefully-selected” axis-aligned random projections of the sample covariance matrix. Additionally, package **RPEnsembleR** ([Cannings and Samworth 2021](#)) implements a similar idea when building the ensemble of classifiers: for each classifier in the ensemble, a collection of (Gaussian, axis-aligned projections, or Haar) random projection matrices is generated, and the one that minimizes a risk measure for classification on a test set is selected. For Python ([Van Rossum et al. 2011](#)) the **sklearn.random_projection** module ([Pedregosa et al. 2011](#)) implements two types of unstructured random matrices, namely Gaussian random matrix and sparse random matrix.

2.3. Generalized linear models

After we perform an initial screening step followed by a projection step in each marginal model, we assume that the reduced and projected set of predictors $\mathbf{z}_i = \Phi \mathbf{x}_i \in \mathbb{R}^m$ together

with the response arise from a GLM with the response having conditional density from a (reproductive) exponential dispersion family of the form

$$f(y_i|\theta_i, \phi) = \exp\left\{\frac{y_i\theta_i - b(\theta_i)}{a(\phi)} + c(y_i, \phi)\right\}, \quad g(E[y_i|\mathbf{z}_i]) = \gamma_0 + (\Phi\mathbf{x}_i)^\top \boldsymbol{\gamma} =: \eta_i,$$

where θ_i is the natural parameter, ϕ is a dispersion parameter, $a(\cdot) > 0$ and $c(\cdot)$ are specific real-valued functions determining different families, and $b(\cdot)$ is the log-partition function normalizing the density to integrate to one. If ϕ is known, we obtain densities in the natural exponential family for our responses. The responses are related to the m -dimensional reduced and projected predictors through the conditional mean, i.e., the conditional mean of y_i given $\mathbf{z}_i$ depends on a linear combination of the predictors through (invertible) link function $g(\cdot)$, where $\gamma_0 \in \mathbb{R}$ is the intercept and $\boldsymbol{\gamma} \in \mathbb{R}^m$ is a vector of regression coefficients for the m projected predictors. We can see that the original coefficients of the predictors $\mathbf{x}_i$ can be obtained as: $\boldsymbol{\beta} = \Phi^\top \boldsymbol{\gamma}$.

Estimates for $\gamma_0 \in \mathbb{R}$ and $\boldsymbol{\gamma} \in \mathbb{R}^m$ can be obtained by directly maximizing the log-likelihood below, as generally one chooses $m < n$:

$$\ell(\beta_0, \boldsymbol{\beta}) = \sum_{i=1}^n \frac{y_i\theta_i(\beta_0, \boldsymbol{\beta}, \mathbf{x}_i) - b(\theta_i(\beta_0, \boldsymbol{\beta}, \mathbf{x}_i))}{a(\phi)} + c(y_i, \phi),$$

where $\theta_i(\beta_0, \boldsymbol{\beta}, \mathbf{x}_i) = (b')^{-1}(g^{-1}(\eta_i))$. However, even if $m < n$ it might still be desirable to add a (e.g., small ℓ_2) penalty to the likelihood of the marginal models—for the binomial family this can alleviate problems related to separation while keeping the bias low. Moreover, if outlier influence should be reduced, M - or trimmed estimators could be employed for obtaining robust estimates of the regression coefficients (see e.g., [Cantoni and Ronchetti 2001](#)).

2.4. SPAR algorithm

We present the general algorithm for sparse projected averaged regression (SPAR) implemented in package **spareg**. A diagram of the algorithm is shown in Figure ??.

1. Choose family with corresponding log-likelihood $\ell(\cdot)$ and link $g(\cdot)$.
2. Standardize the $(n \times p)$ matrix of predictors X for all families and the vector of responses $\mathbf{y}$ for the Gaussian family by subtracting the sample mean and dividing by the sample standard deviation of each variable.
3. Calculate screening coefficients $\hat{\boldsymbol{\omega}}$.
4. For $k = 1, \dots, M$ models:
 - (a) If $p > p_0$, where p_0 is the number of variables to be screened, screen p_0 predictors based on the screening coefficients $\hat{\boldsymbol{\omega}}$, which yields for model k the screening index set $I_k = \{j_1^k, \dots, j_{p_0}^k\} \subset \{1, \dots, p\}$. If probabilistic screening should be employed, draw the predictors sequentially without replacement using an initial vector of probabilities $\boldsymbol{\pi}$ with $\pi_j \propto |\hat{\omega}_j|$; otherwise, select the p_0 variables with the largest $|\hat{\omega}_j|$. If $p \leq p_0$, perform no screening and set $I_k = \{1, \dots, p\}$.

- (b) Project the screened variables to a randomly drawn dimension $m_k \sim \text{Unif}(\{m_l, \dots, m_u\})$ using *projection matrix* Φ_k to obtain $Z_k = X_{\cdot I_k} \Phi_k^\top \in \mathbb{R}^{n \times m_k}$, where $X_{\cdot I_k}$ contains the columns in X having a column index in I_k .
 - (c) Fit a *GLM*-type model of $\mathbf{y}$ on Z_k to obtain estimated coefficients $\hat{\gamma}^k \in \mathbb{R}^{m_k}$ and $\hat{\beta}_{I_k}^k = \Phi_k^\top \hat{\gamma}^k$, $\hat{\beta}_{\bar{I}_k}^k = 0$, where $\bar{I}_k = \{1, \dots, p\} \setminus I_k$.
5. For a given threshold $\nu \geq 0$, set all $\hat{\beta}_j^k$ with $|\hat{\beta}_j^k| \leq \nu$ to 0 for all j, k .
 6. Choose M and ν via a validation set or cross-validation evaluating a two-dimensional grid of values. If cross-validation is performed, repeat Steps 2 to 6 for each training fold using the fixed index set I_k and projection matrix Φ_k , and assess the performance on the test fold by a loss function $K(M, \nu)$:
$$(M_{\text{best}}, \nu_{\text{best}}) = \operatorname{argmin}_{M, \nu} K(M, \nu).$$
 7. Combine models of the ensembles either via the coefficients by using a *simple average* of the regression coefficients $\hat{\beta} = \sum_{k=1}^M \hat{\beta}^k / M$ (this results in averaging of the models on the *link level*) or via the fitted values by taking a *simple average* on the *response level* i.e., $\hat{y} = \sum_{k=1}^M g^{-1}(\mathbf{x}_i^\top \hat{\beta}^k) / M$ (here we omit the intercept for the sake of notational simplicity).
 8. Output the averaged estimated coefficients and predictions for $(M_{\text{best}}, \nu_{\text{best}})$.

The proposed algorithm is flexible, allowing for various choices of screening coefficients $\hat{\mathbf{w}}$, random projection matrices Φ and marginal models. Several variants have been introduced in the literature. For example, [Mukhopadhyay and Dunson \(2020\)](#) propose a version that uses probabilistic marginal correlation screening along with the sparse random projection method from [Achlioptas \(2003\)](#), without applying thresholding. More recent work by [Parzer et al. \(2025, 2024\)](#) extends this approach by employing data-informed random projection that leverages information from the ridge coefficients and complement the projection step with probabilistic screening based on the same ridge regression coefficients (see details in Section 2.2).

In order to choose default values for p_0 , m_l , m_u as well as the maximal number of considered models M , we rely on previous literature. Regarding the number of screened variables, [Parzer et al. \(2025\)](#) propose choosing $p_0 = c \cdot n$ as a multiple of n (thus independent of p) and find that the best results are achieved for $2 \leq c \leq 4$. We thus set in the algorithm $p_0 = 2n$ as a default value.

The dimension m_k is random for the purpose of introducing more variability in the ensemble and reducing the reliance on a fixed (possibly arbitrarily chosen) goal dimension. [Mukhopadhyay and Dunson \(2020\)](#) propose using $m_l = 2 \log(p)$ and $m_u = 3n/4$ while [Parzer et al. \(2024\)](#) use slightly smaller goal dimensions ($m_l = \log(p)$, $m_u = n/2$), to reduce the dimension of the marginal models to be estimated. On the other hand, for random projection matrices satisfying the JL lemma, m_l can be derived from the theoretical bounds for preserving the distances between all pairs of points approximately. We employ as default values the ones proposed in [Parzer et al. \(2025\)](#).

A further modification of the algorithm above is to employ part of the data for estimating the screening coefficients (Step 3) and the remaining data for estimating the ensemble models

(Step 4). Such a strategy could avoid issues related to overfitting. Even though [Parzer et al. \(2025\)](#) find that in the linear regression case such a splitting approach does not improve performance, the implementation in **spareg** allows for data splitting.

[Parzer et al. \(2025\)](#) find that, when using their proposed data-driven random projection and HOLP screening coefficients in a linear regression setting, the gain in predictive performance when using more than $M = 20$ models is marginal. We use therefore $M = 20$ as a default value in the algorithm. However, for ultra high-dimensional settings, larger values of M can be beneficial to explore more of the predictor space, particularly when using data-agnostic projections. In such cases, stronger thresholding (higher ν) is also recommended to mitigate overfitting, especially when no screening is applied or when the expected sparsity is medium to high. When using data-agnostic projections, we encourage users to explore a grid of M up to 50 or 100 to encourage diversity in the ensemble.

The default grid for ν , based on equally spaced quantiles of the absolute values of the estimated coefficients, generally yields reasonable performance. However, in high-dimensional settings with high expected sparsity, thresholding becomes particularly useful for larger n values and especially when employing the CW data-dependent projection with ridge coefficients, as it helps identify zero coefficients in the recovered signal.

Finally, when choosing the optimal M and ν in Step 6 given a grid of values for each parameter, one can pick the combination delivering the lowest loss function or the combination delivering the sparsest model with a loss within one standard deviation of the minimum (this corresponds to the commonly employed “one-standard-error (1se)” rule in penalized regression). We note that the cross-validation on the grid of number of models is not computationally intensive as we only need to train the maximum number of models considered in the grid on each training fold and evaluate performance by aggregating the first M_1 models, then the first M_2 , and so on.

3. Software

Package **spareg** is available on CRAN and can be installed and loaded as follows:

```
R> install.packages("spareg")
R> library("spareg")
```

An overview of the main functions is provided in Table 1, followed by detailed descriptions in the remainder of this section. For demonstration, we use a simulated dataset with a continuous response and $p = 2000$ predictors where $n = 200$ observations are used as training and $n = 100$ observations are used as test data. The data is generated using `simulate_spareg_data()`, which simulates by default a linear regression model with $\sigma^2 = 83$, intercept $\mu = 1$ and coefficients β with 100 non-zero entries uniformly sampled from $\{-3, -2, -1, 1, 2, 3\}$:

```
R> set.seed(1234)
R> example_data <- simulate_spareg_data(n = 200, p = 2000, ntest = 200)
R> str(example_data)
```

List of 7

```
$ x      : num [1:200, 1:2000] 0.0315 1.002 0.3063 -0.0661 0.0577 ...
```

```
$ y      : num [1:200] -11.88 2.23 -9.34 -14.22 -5.97 ...
$ xtest  : num [1:200, 1:2000] -0.386 0.49 1.676 -0.151 0.239 ...
$ ytest  : num [1:200] 24.7 -7.24 40.59 37.98 24.09 ...
[list output truncated]
```

3.1. Main functions and their arguments

The two main functions for fitting the SPAR algorithm are:

- `spar()` fits the SPAR algorithm without cross-validation and returns an ‘spar’ object.
- `spar.cv()` implements the cross-validated procedure and returns a ‘spar.cv’ object.

Additionally, the aliases `spareg()` and `spareg.cv()` are available for convenience. Both functions accept the following key arguments:

- `x`—an $n \times p$ numeric matrix of predictors and `y`—a numeric vector of length n .
- `family`—an object from `stats::family()`; defaults to `gaussian()`.
- `model`—an object of class ‘sparmodel’ specifying the model for each element of the ensemble. Defaults to `spar_glm()` for Gaussian-identity and to `spar_glmnet()` for others.
- `rp`—an object of class ‘randomprojection’. Defaults to `rp_cw(data = TRUE)`.
- `screencoef`—an object of class ‘screencoef’. Defaults to `NULL` (no screening).
- `nnu`—number of threshold values ν to consider. Defaults to 20.
- `nus`—an optional vector of threshold values ν . If not provided, defaults to a grid of `nnu` values: zero and `nnu - 1` equally spaced quantiles of the absolute values of the non-zero marginal coefficients.
- `nummods`—number of models in the ensemble. Defaults to 20. Also allows for a vector of ensemble sizes.
- `measure`—performance measure for selecting M and ν . Options: “deviance” (default), “mse” (mean squared error), “mae” (mean absolute error), “class” (misclassification error), or “1-auc” (one minus area under the ROC curve).
- `avg_type`—stage of aggregation for computing the performance measure—either on “link” or “response” level.
- `parallel`—use parallel backend if available. Defaults to `FALSE`.

Furthermore, `spar()` has the arguments `xval` and `yval`—validation data for choosing M_{best} and ν_{best} (if not provided, `x` and `y` will be employed). Optional argument `inds` can be used to pass a list specifying columns of variables to retain in each marginal model while projection

Function	Description
<i>High-level functions—return S3 objects of class ‘spar’ and ‘spar.cv’</i>	
spar()	Fits the ensemble. Supports parallelization and validation-set tuning for M and ν .
spar.cv()	Cross-validated wrapper for optimal M and ν selection.
<i>Screening functions screen_*(..., control = list())—return S3 objects of class ‘screencoef’</i>	
<i>Common attributes to be passed by ...</i>	
nscreen :	Variables to retain. <i>Default</i> : $\min(2n, p)$
split_data_prop :	Proportion of data for screening. <i>Default</i> : 1.0 (full)
type :	Screening logic (probabilistic "prob" or "fixed"). <i>Default</i> : "prob"
screen_marglik()	Univariate marginal likelihood screening using stats::glm() .
screen_cor()	Correlation-based screening using stats::cor ; accepts method for correlation type in control argument. <i>Default</i> : method = "pearson".
screen_glmnet()	Multivariate screening using glmnet::glmnet and its controls. <i>Default</i> : elastic net mixing parameter alpha = 0 (ridge regression).
<i>Random projection functions rp_*(..., control = list())—return S3 objects of class ‘randomprojection’</i>	
<i>Common attributes to be passed by ...</i>	
mslow :	Minimum projection dimension. <i>Default</i> : $\log(p)$
msup :	Maximum projection dimension. <i>Default</i> : $\min(n/2, p)$
rp_gaussian()	Projection with iid normal entries using stats::rnorm() . <i>Default</i> : mean = 0, sd = 1.
rp_sparse()	Sparse projection with entries from the Rademacher distribution (parameter psi passed through control). <i>Default</i> : psi = 1.
rp_cw() [†]	Sparse embedding matrix. <i>Default</i> : rp_cw(data = TRUE) for data-driven diagonals, otherwise randomly generated.
<i>Marginal model functions spar_*(..., control = list())—return S3 objects of class ‘sparmodel’</i>	
spar_glm()	Fits unregularized GLM for each projected model using stats::glm ; accepts glm controls in control argument).
spar_glmnet() [†]	Fits marginal elastic net models using glmnet::glmnet ; accepts glmnet controls in control argument). <i>Default</i> : small ridge penalty.
<i>S3 methods for ‘spar’ and ‘spar.cv’ objects</i>	
coef()	Extracts raw or aggregated coefficients as an S3 ‘coefspar’ object for specific (M, ν) . <i>Default</i> : mean aggregation for best (M, ν) .
predict()	Generates predictions aggregated on "link" or "response" levels for specific (M, ν) . <i>Default</i> : link-level mean aggregation for best (M, ν) .
plot()	Returns a ‘ggplot’ of (cross-)validation measure, sparsity, residuals or raw coefficients. <i>Default</i> : (cross-)validation measure as a function of ν .
<i>Extractor and constructor functions</i>	
get_intercept()	Extracts intercept(s) from a ‘coefspar’ object.
get_coef()	Extracts the (aggregated) coefficients from ‘coefspar’ object.
get_model()	Retrieves the (M, ν) model minimizing the (cross-)validation measure; supports one-standard-error choice for ‘spar.cv’.
get_measure()	Extracts performance measure and sparsity statistics across the tuning grid from ‘spar’ and ‘spar.cv’ objects.
constructor_* (...)	Helpers to create ‘screencoef’, ‘randomprojection’, or ‘sparmodel’ objects.

Table 1: Summary of key functions in the **spareg** package. [†] marks the modular components used as defaults in the **spar()** and **spar.cv()** functions. By default, no screening is performed.

matrices to be used in each marginal model can be passed as a list through optional argument

RPMs.

Function `spar.cv()` has specific arguments `nfolds`, the number of folds to be used (defaults to 10) and `precompute_mode = c("precompute_all", "precompute_rpm", "full_cv")`. Argument `precompute_mode` indicates whether to use the same random projection matrix and/or the same indices for screening across folds (defaults to `"precompute_all"`). If one of the first two options is selected, `spar.cv()` uses a lightweight version of `spar()` in each fold. Here, random projection matrices and/or screening indicators are computed once on the full dataset before cross-validation. This is achieved by an initial call to `spar()` on the entire dataset. If `"precompute_all"`, screening indicators remain fixed throughout the analysis. The random elements of the projection matrices are also held fixed in all folds for `"precompute_all"` and `"precompute_rpm"`. However, if data-driven projections are used, the training data for each fold will be used to update the data-dependent entries of the projection matrix in all cases. If `precompute_mode = "full_cv"` is selected, the random projection and screening indicators are recomputed in each fold. Fixing the screening indicators and/or (random elements of) projection matrices prior to cross-validation reduces computational overhead in some settings. On the other hand, fixing these components removes a source of variability from the cross-validation process which can lead to a more stable and interpretable selection of M and ν , as the cross-validation results are not confounded by variations in the screening or projection steps. However, this approach can be affected by target leakage, as estimates obtained on the whole data are used in the cross-validation exercise. This is particularly relevant when performing screening. To avoid target leakage `precompute_mode = "full_cv"` or `precompute_mode = "precompute_rpm"` can be selected.

3.2. Extensible S3 objects

The `spar()` framework is organized around a common extensible structure comprising three method-specification objects: `'screencoef'` for screening coefficients, `'randomprojection'` for random projection matrices and `sparmodel` for the marginal model to be used in the ensemble.

We will explain in the following how each object is created to store the selected method and its initial `control` parameters. These objects are created before data is passed to `spar()`. Once the relevant training data and response family are available, `spar()` invokes the object's `update_fun()` to resolve context- or data-dependent settings, adjust controls, and store any required state; it then invokes the corresponding generation function `generate_fun()` to perform the primary computation. As a convention, `control` is a list of parameters to be passed to the main generating function, while parameters related to the procedure itself (such as the number of variables to be screened or the minimum dimension to which the variables should be projected) are to be saved as attributes to the objects. The shared interfaces allow users to extend the framework with custom screening, projection, and modeling methods while preserving consistent behavior across the three object types.

Screening coefficients

Screening coefficients in `spareg` are implemented as S3 classes `'screencoef'`. These objects are created using functions like `screen_*` which accept as arguments: `...` (i.e., ellipsis) and `control` (a list of parameters for computing the screening coefficients). Additional arguments passed through the ellipsis will be saved as attributes of the object.

The following screening coefficients are implemented in **spareg**:

- **screen_marglik()**—computes the screening coefficients by the coefficient of $\mathbf{X}_j$ for $j = 1, \dots, p$ in a univariate GLM using the **stats::glm()** function. It allows to pass a list of controls through the **control** argument to **stats::glm()** e.g., **screen_marglik(control = list(family = binomial(probit)))**.
- **screen_cor()**—computes the screening coefficients by the correlation between $\mathbf{y}$ and $\mathbf{X}_j$ using the function **stats::cor()**. It allows to pass a list of controls through the **control** argument to **stats::cor()**—e.g., **screen_cor(control = list(method = "spearman"))**.
- **screen_glmnet()**—computes by default a ridge coefficient with a small penalty:

$$\hat{\omega} := \operatorname{argmin}_{\beta \in \mathbb{R}^p} \min_{\beta_0 \in \mathbb{R}} \sum_{i=1}^n -\ell(\beta; y_i, \mathbf{x}_i) + \frac{\varepsilon}{2} \sum_{j=1}^p \beta_j^2, \varepsilon > 0.$$

This function uses **glmnet()** of package **glmnet** (Friedman, Hastie, and Tibshirani 2010; Tay, Narasimhan, and Hastie 2023) with elastic net mixing parameter $\alpha = 0$ (ridge penalty), a small **lambda.min.ratio** (smallest penalty as a fraction of the maximum value in the grid) and penalty selection based on Parzer *et al.* (2024): the smallest penalty where the deviance ratio is < 0.99 (Gaussian) or < 0.8 (other families). Custom controls can be passed via the **control** argument e.g., **screen_glmnet(control = list(alpha = 0.5))**.

The following attributes can be passed to the **screen_***() function through ...:

- **nscreen**—number of variables to be retained after screening; defaults to $2n$. If $2n > p$, no screening is performed.
- **split_data_prop**—proportion of the data to be used for computing the screening coefficient. The remaining data will be used for estimating the marginal models; default is to use the whole data for screening coefficients and the marginal model estimation.
- **type**—either "prob" (for probabilistic screening—default) or "fixed" (for using a fixed set of **nscreen** variables).
- **reuse_in_rp**—indicates whether the screening coefficients should be reused at a later stage in the construction of a data-dependent random projection. Defaults to **FALSE**. This attribute is ignored if a data-agnostic random projection is used.

All implemented **screen_***() functions return an object of class 'screencoef' which in turn is a list containing an **update_fun** function which in the default case updates the object with the **family** information, a **generate_fun** function, the **control** list passed by the user in **screen_***(), and an optional **name** string. A user-friendly **print()** method of the 'screencoef' object is also provided. For convenience, **constructor_screencoef()**, a constructor function is provided, which can be used to create new **screen_***() functions. An example is presented in Section 4.1.

Random projections

Random projection objects are implemented as S3 classes ‘`randomprojection`’ and are created by functions like `rp_*`(). These functions take the ellipsis `...` and a list of controls `control` as arguments. The package includes the following random projections:

- `rp_gaussian()`—entries are iid from the normal distribution (defaults to standard normal).
- `rp_sparse()`—sparse matrix as in Achlioptas (2003); ψ can be specified via `control` e.g., `rp_sparse(control = list(psi = 1/3))`. Defaults to `psi = 1`.
- `rp_cw()`—sparse embedding random projection in Clarkson and Woodruff (2013). If the random projection is specified as `rp_cw(data = TRUE)` and the screening coefficient object has the attribute `reuse_in_rp = TRUE`, the diagonal entries of D in Equation 1 are set to the screening coefficients. Otherwise, if `rp_cw(data = TRUE)`, the data-driven modification to the CW matrix proposed in Parzer *et al.* (2024) is used.

Common attributes which can be passed via `...` to the `rp_*`() functions include:

- `mslow`—minimum dimension for projecting the predictors; defaults to $\log(p)$. Note that for JL-type random projections, `mslow` can be chosen from JL bounds.
- `msup`—maximum dimension for projecting the predictors; defaults to $n/2$ or, if $p < n/2$, to p .

The `rp_*`() functions return an object of class ‘`randomprojection`’ which is a list of five elements. A constructor function `constructor_randomprojection()` for creating these objects is provided (as will be shown in Section 4.2). The most important elements contain `generate_fun()`, the function for generating the random projection matrix, and `control`, the list of controls passed by the user in `rp_*`(). As previously mentioned, objects of class ‘`randomprojection`’ are constructed before the data are available. Consequently, settings that depend on the data cannot always be resolved by the constructor. Before calling `generate_fun()`, `spar()` calls `update_fun` to resolve such settings and to store any (data-dependent) information in the object (such as ‘`family`’ or precomputed screening coefficients). If argument `update_fun` is not specified when using a constructor, the default will only add the family information to the ‘`randomprojection`’ object. Further optional elements include `name`, or function `update_rpm_w_data()`, which specifies how to update the pre-computed matrices (when `precompute_mode` is `precompute_all` or `precompute_rpm`) with data information in each fold of the cross-validation procedure. Note that `update_fun()` updates the RP specification before generating matrices while `update_rpm_w_data()` modifies an already-generated matrix, usually during cross-validation.

Marginal models

The `spareg` package uses the ‘`sparmodel`’ class to define marginal models for each ensemble element. These models assume a linear predictor (a linear combination of projected variables). Functions `spar_*`() create these objects and have arguments `...` (i.e., the ellipsis) and `control` (list used in the main function for building the model). All arguments passed through the ellipsis will be saved as attributes of the object.

The two functions implemented are

- `spar_glm()`—fits unregularized GLMs using `stats::glm()` (default for Gaussian family with identity link).
- `spar_glmnet()`—fits regularized GLMs using function `glmnet::glmnet()` (default for all other families using $\alpha = 0$ with the minimal penalty considered in `glmnet`¹).

By default, `spar_glm()` is used for Gaussian family with identity link, while `spar_glmnet()` is used for all other families.

Similar to the already presented S3 objects, an object of class ‘`sparmodel`’ is a list containing an optional character `name` for printing, `generate_fun()`—a function used to generate the coefficients and intercept value for each model in the ensemble, `update_fun()`—an (optional) function to modify the ‘`sparmodel`’ object at the start of the SPAR algorithm (e.g., adjusting the ‘`family`’ object for `glmnet` compatibility) and `control`, the list of controls passed by the user in `spar_*`().

An example for implementing a new marginal model class using the constructor function `constructor_sparmodel()` is presented in Section 4.3.

3.3. Methods

Methods `print()`, `plot()`, `coef()`, `predict()` are available for both ‘`spar`’ and ‘`spar.cv`’ classes.

`print()`

The `print()` method returns information on M_{best} and ν_{best} , the number of active predictors (i.e., predictors which have at least a non-zero coefficient across the marginal models) and a summary of the non-zero non-standardized coefficients. For ‘`spar.cv`’ it shows information for the (M, ν) combination chosen by the best and the 1se rule.

```
R> set.seed(12)
R> spar_res <- spar(example_data$x, example_data$y,
+   xval = example_data$xtest, yval = example_data$ytest,
+   nummods = c(5, 10, 15, 20, 25, 30))
R> spar_res
```

`spar` object:

Smallest validation measure (deviance) of 5.97e+04 reached for nummod=10,
nu=1.6464e-03 leading to 1612 / 2000 active predictors.

Summary of those non-zero coefficients:

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
-0.69473	-0.13242	0.06111	0.02073	0.17086	0.82134

¹In package `glmnet` $\lambda_{\min} = \lambda_{\max}/100$ where $\lambda_{\max}$ is the smallest value that shrinks all coefficients to zero and is calculated as the maximum absolute inner product of the standardized predictors and the scores evaluated at the null model, divided by the number of observations times α . For ridge regression $\alpha = 0.001$ is used for computational purposes.

`coef()`

Method `coef()` extracts coefficients with arguments:

- `nu, nummod`—threshold values and number of marginal models to use in the calculation of the coefficients. Defaults to the values with minimal validation error.
- `aggregate`—aggregation method for coefficients across the `nummod` ensemble models. Options: "mean" (default), "median" or "none" (skips the aggregation and returns the matrix of raw coefficients for each marginal model—useful for understanding the variability of the coefficients across the ensemble models).
- `opt_par`—only for 'spar.cv' with `precompute_mode = "precompute_all"`. In this case, it chooses between "best" or "1se". Otherwise, coefficients cannot be generated without refitting on whole data with "best" or "1se" parameters by calling `spar()` with the desired (M, ν) combination.

Method `coef()` returns an object of class 'coefspar', for which user-friendly `print()` and `summary()` methods as well as extractor functions `get_intercept()` and `get_coef()` are provided.

```
R> coef(spar_res)
```

Coefficients from spar object:

Based on the *best rule* (min error) selection

Aggregation method over models: mean

Selected combination: 10 models, threshold = 0.001646391

Coefficients:

(Intercept)	V1	V2	V3	V4
2.2976	0.0595	-0.1569	0.1286	0.1781

... (1996 coefficients not shown)

Number of active variables: 1612/2000

```
R> get_intercept(coef(spar_res, nummod = 5, aggregate = "none"))
```

Model_1	Model_2	Model_3	Model_4	Model_5
2.262267	2.275260	2.339915	2.378123	2.253901

`predict()`

Functionality for computing predictions is provided through the method `predict()` with arguments:

- `xnew`—new predictor matrix.
- `type`—the type of predictions, either on "response" level (default) or on "link" level.

- `avg_type`—stage of aggregation: at the level of regression coefficients (i.e., on the `link` level) or at the fitted values level (i.e., `response` level).
- `nu`, `nummod`, — threshold values and number of models used to compute the averaged coefficients; defaults to optimal values.
- `aggregate`—aggregation method ("`mean`" or "`median`"). Defaults to mean aggregation.
- `opt_par`—only for `'spar.cv'` objects fitted with `precompute_mode = "precompute_all"`. In this case, it chooses between "`best`" or "`lse`". Otherwise, predictions cannot be generated without refitting on whole data with "`best`" or "`lse`" parameters. To generate predictions, `spar()` can be called to refit the ensembles with the desired (M, ν) combination.

plot()

Plotting functionality is provided through the `plot()` method with arguments:

- `plot_type`—one of:
 - "`val_measure`" plots the (cross-)validation `measure` for either a grid of `nu` values for a fixed number of models `nummod` or vice versa.
 - "`val_numactive`" plots the number of active variables for either a grid of `nu` values for a fixed number of models `nummod` or vice versa.
 - "`res_vs_fitted`" produces a residuals-vs-fitted plot. The residuals are computed as $r_i = y_i - \hat{y}_i$, where $\hat{y}_i$ is the prediction computed on response level (available for `'spar'` objects and `'spar.cv'` object with `precompute_mode = "precompute_all"`).
 - "`coefs`" produces a plot of the standardized coefficient values for each predictor in each marginal model (before thresholding). For each predictor, the coefficients are sorted from largest to smallest in absolute value across the marginal models and their values are represented in the plot using a color scale (available for `'spar'` objects and `'spar.cv'` object with `precompute_mode = "precompute_all"`).
- `plot_along`—for `plot_type = "val_measure"` or "`val_numactive`" only, it indicates whether to vary `nu` or `nummod` on the x axis.
- `nummod`—fixed value for number of models when `plot_along = "nu"` for `plot_type = "val_measure"` or "`val_numactive`"; if `plot_type = "res_vs_fitted"`, it is used in `predict()`.
- `nu`—fixed value for ν when `plot_along = "nummod"` for `plot_type = "val_measure"` or "`val_numactive`"; if `plot_type = "res_vs_fitted"`, it is used in `predict()`.
- `xfit` and `yfit`—predictor matrix and response vector for `plot_type = "res_vs_fitted"`.
- `prange`, `coef_order`—only for `plot_type = "coefs"`, optional customization parameters. Use `prange` to select a specific range of predictors to display, and `coef_order` to rearrange their order.

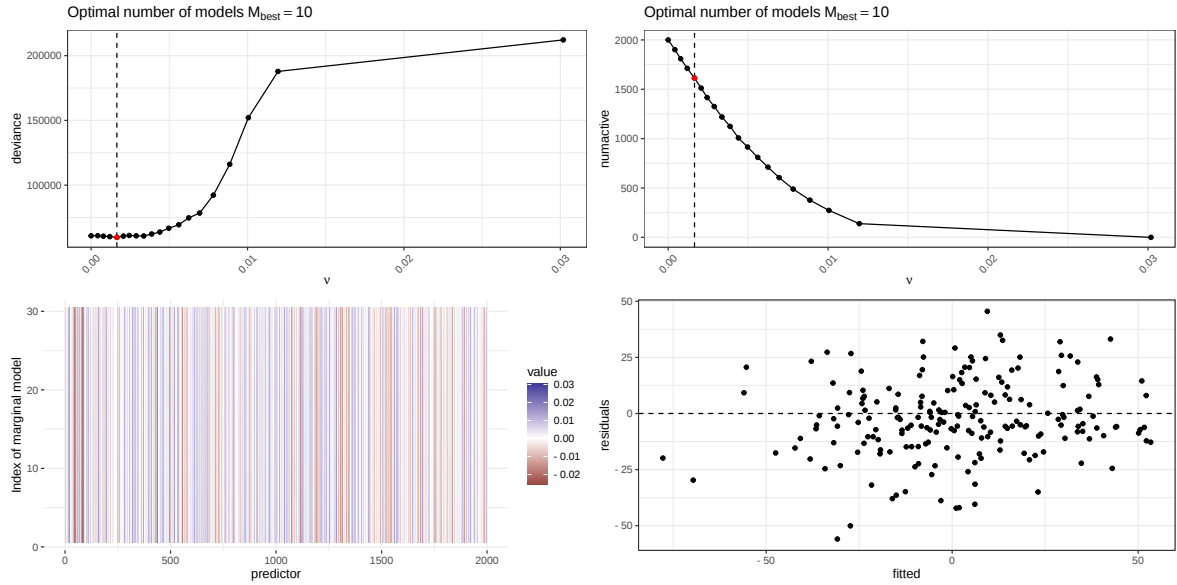


Figure 1: `plot()` methods for ‘`spar`’ object. They refer to the `plot_type` of “`val_measure`” (top left), “`val_numactive`” (top right), “`coefs`” (bottom left), and “`res_vs_fitted`” (bottom right), respectively. The red dots in the top figures show the best ν chosen on the validation set for the optimal number of models.

- `opt_par`—only for `plot_type = "res_vs_fitted"` and ‘`spar.cv`’ objects fitted with `precompute_mode = "precompute_all"`. It indicates whether predictions should be computed using the “`best`” or “`1se`” selection rule.

The `plot()` methods return objects of class ‘`ggplot`’ (Wickham 2016). The four types of plots described above for a ‘`spar`’ object are shown in Figure 1.

3.4. Parallelization

The package supports parallelization in the estimation of the ensemble models via package `foreach` (Microsoft and Weston 2022). This is possible by setting the parameter `parallel = TRUE` in `spar()` or `spar.cv()`, assuming that a parallel backend for `foreach` is registered using the `registerDoParallel()` function of package `doParallel` (Microsoft Corporation and Weston 2022). To ensure reproducibility the `doRNG` (Gaujoux 2025) can be used. Our experiments suggest that parallelization pays off especially for data sets with a larger number of observations n . A minimal example with a correlation-based probabilistic screening and a Gaussian random projection matrix is provided below:

```
R> ex4 <- simulate_spareg_data(n = 1000, p = 2000, ntest = 1000, seed = 123)
R> library(doParallel)
R> library(doRNG)
R> cl <- makeCluster(2, type = "PSOCK")
R> registerDoParallel(cl)
R> registerDoRNG(seed = 123)
R> spar_res_par <- spar(ex4$x, ex4$y, screencoef = screen_cor(),
```

```
+   rp = rp_gaussian(), nummods = 50, parallel = TRUE)
R> stopCluster(cl)
```

4. Extensibility

4.1. Screening coefficients

We demonstrate how screening coefficients implemented in package **VariableScreening** can easily be incorporated in the framework of **spareg**.

We start by defining the function for generating the screening coefficients using function `screenIID()` in **VariableScreening**.

```
R> generate_scr_sirs <- function(object, x, y, ...) {
+   ctrl <- object$control[names(object$control) %in%
+                         names(formals(VariableScreening::screenIID)))]
+   res_screen <- do.call(function(...)
+     VariableScreening::screenIID(x, y, ...), ctrl)
+   res_screen$measurement
+ }
```

Note that `screenIID()` also takes `method` as an argument. To allow for flexibility, we do not fix the screening method in `generate_scr_sirs()` but rather allow the user to pass a method through the `control` argument in the `screen_*`() function. This function is created using the helper `constructor_screencoef()`:

```
R> screen_sirs <- constructor_screencoef(generate_fun = generate_scr_sirs,
+   name = "screen_sirs")
```

We now call `spar()` with the newly created screening procedure. We consider the method SIRS of [Zhu et al. \(2011\)](#), which ranks the predictors by their correlation with the rank-ordered response; we do not perform probabilistic variable screening but employ the top $2n$ variables in each marginal model. The employed random projection is `rp_sparse()` where we set $\psi = 1/\sqrt{p}$, as proposed by [Li et al. \(2006\)](#).

```
R> set.seed(123)
R> spar_example <- spar(example_data$x, example_data$y,
+   screencoef = screen_sirs(type = "fixed", control = list(method = "SIRS")),
+   rp = rp_sparse(psi = 1/sqrt(ncol(example_data$x))), measure = "mse")
```

4.2. Random projections

To extend the package, we incorporate the random projection method proposed by [Cannings and Samworth \(2017\)](#), which uses the Haar measure to generate orthogonal matrices. This method involves drawing entries from a standard normal distribution and applying QR decomposition.

```
R> simulate_haar <- function(m, p) {
+   R0 <- matrix(1/sqrt(p) * rnorm(p * m), nrow = p, ncol = m)
+   t(qr.Q(qr(R0), complete = FALSE))
+ }
```

Cannings and Samworth (2017) propose selecting “good” random projections to optimize out-of-sample prediction. Specifically, for each of the models in the ensemble, B Haar random projections are generated. The projection with the lowest error on a test set (a random subset of proportion ξ of the data) is selected for projecting the variables. While the B projections are data-agnostic, the entire dataset is required to identify the best projection.

In the implementation, we fit a GLM as a marginal model on the training data. The performance is evaluated using deviance on the test set. Default values can be specified in `update_fun()`, which must have arguments `object`, `x` and `y` (predictor matrix and response vector already standardized and supplied by `spar()`), `family` and `...`, whereas all other relevant arguments from `spar()` will be passed through the ellipsis.

```
R> update_rp_cannings <- function(object, x, y, family, ...) {
+   if (is.null(object$control$alpha)) object$control$alpha <- 0
+   if (is.null(object$control$B)) object$control$B <- 50
+   if (is.null(object$control$xi)) object$control$xi <- 0.25
+   if (is.null(object$control$family)) object$control$family <- family
+   object
+ }
```

Next we specify the function for generating the random projection.

```
R> generate_cannings <- function(object, x, y, m, included_vector, ...) {
+   xs <- x[, included_vector]; n <- nrow(x); p <- ncol(xs)
+   id_test <- sample(n, size = n * object$control$xi)
+   xtrain <- xs[-id_test, ]; xtest <- xs[id_test,]
+   ytrain <- y[-id_test]; ytest <- y[id_test]
+   control_glm <-
+     object$control[names(object$control) %in% names(formals(stats::glm.fit))]
+   best_val <- Inf
+   for (b in seq_len(object$control$B)) {
+     RM <- simulate_haar(m, p)
+     xrp <- tcrossprod(xtrain, RM)
+     mod <- do.call(function(...)
+       glm.fit(x = cbind(1, xrp), y = ytrain, ...), control_glm)
+     eta_test <- cbind(1, tcrossprod(xtest, RM)) %*% coef(mod)
+     pred <- object$control$family$linkinv(eta_test)
+     out_dev <- sum(object$control$family$dev.resid(ytest, pred, wt = 1))
+     if (out_dev < best_val) {
+       best_val <- out_dev; best_RM <- RM
+     }
+   }
+   return(best_RM)
+ }
```

Putting it all together, we get:

```
R> rp_cannings <- constructor_randomprojection(
+   name = "RP Cannings and Samworth 2017",
+   generate_fun = generate_cannings, update_fun = update_rp_cannings)
```

We can now estimate SPAR for a binomial model, after simulating data from a linear model and then transforming the response to a binary scale:

```
R> set.seed(1234)
R> ex2 <- simulate_spareg_data(n = 200, p = 500, ntest = 100)
R> ys <- (ex2$y > 0) + 0; ysval <- (ex2$ytest > 0) + 0
```

We use only one value for the number of models $M = 50$ (which is in line to recommendations in [Cannings and Samworth 2017](#)), no screening and no thresholding. We set the maximum dimension for the random projection to $m_u = 20$ to avoid generating large matrices and for stability in the marginal binomial GLMs.

```
R> set.seed(1234)
R> spar_cs <- spar(x = ex2$x, y = ys, family = binomial(),
+   rp = rp_cannings(msup = 20), model = spar_glm(),
+   nus = 0, nummods = 50, xval = ex2$xtest, yval = ysval)
```

We can compare with results obtained by using the data-agnostic sparse embedding matrix `rp_cw(data = FALSE)`:

```
R> set.seed(1234)
R> spar_cw <- spar(x = ex2$x, y = ys, family = binomial(),
+   rp = rp_cw(data = FALSE, msup = 20), model = spar_glm(),
+   nus = 0, nummods = 50, xval = ex2$xtest, yval = ysval)
```

The two approaches are compared by directly looking at the validation measure achieved by the two ensembles on the provided validation set. The extractor function `get_measure()` can be used to extract the following information from the ‘spar’ object:

```
R> rbind("cannings" = get_measure(spar_cs), "cw" = get_measure(spar_cw))
```

	nu	nummod	deviance	numactive
cannings	0	50	82.96547	500
cw	0	50	85.07831	500

4.3. Marginal models

Finally, we extend the package by implementing a robust GLM as a marginal model using the package **robustbase** ([Todorov and Filzmoser 2009](#)).

We define a `generate_fun()` that uses `robustbase::lmrob()` for Gaussian models with identity link and `robustbase::glmrob()` for other family-link combinations. The function

must take by design the arguments `object` (of class ‘`sparmodel`’), `x` and `y` (standardized predictors and response vector passed by `spar()`), `z` (matrix of reduced predictors) as well as It outputs a list with two elements: `gammas` (vector of coefficients) and the `intercept` value.

```
R> model_glmrob <- function(object, x, y, z, ...) {
+   requireNamespace("robustbase")
+   fam <- object$control$family
+   f <- ifelse(fam$family == "gaussian" & fam$link == "identity",
+     robustbase::lmrob, robustbase::glmrob)
+   glmrob_res <- do.call(function(...)
+     f(y ~ as.matrix(z), ...), object$control)
+   cfs <- coef(glmrob_res)
+   list(gammas = cfs[-1], intercept = cfs[1])
+ }
```

Function `spar_glmrob()` that builds the ‘`sparmodel`’ object can be created by:

```
R> spar_glmrob <- constructor_sparmodel(generate_fun = model_glmrob)
```

We demonstrate this on simulated Poisson data with 25% contaminated predictors:

```
R> set.seed(123)
R> example_data3 <- simulate_spareg_data(n = 100, p = 1000,
+   ntest = 100, snr = 10, beta_vals = c(-1, 1)/10)
R> ypois <- round(exp(example_data3$y)) + 1; x <- example_data3$x;
R> np <- ncol(x) * nrow(x)
R> id_outliers_x <- sample(seq_len(np), 0.25 * np)
R> x[id_outliers_x] <- x[id_outliers_x] + 50
```

We then compare standard GLMs to SPAR with robust GLM marginal models, using no screening, `rp_gaussian(msup = 25)`—instead of the default `msup` of $n/2 = 50$ to improve stability—and MAE as the selection criterion.

```
R> set.seed(1234)
R> spar_rob_res <- spar(x, ypois, family = poisson(),
+   model = spar_glmrob(), rp = rp_gaussian(msup = 25), measure = "mae")
R> set.seed(1234)
R> spar_res <- spar(x, ypois, family = poisson(),
+   model = spar_glm(), rp = rp_gaussian(msup = 25), measure = "mae")
```

The `get_model()` function extracts the optimal ensemble from a ‘`spar`’ or ‘`spar.cv`’ object. It selects either the best model (`opt_par = "best"`, default) or, for ‘`spar.cv`’ objects, the model based on the 1se rule (`opt_par = "1se"`).

```
R> best_rob <- get_model(spar_rob_res, opt_par = "best")
R> best_glm <- get_model(spar_res, opt_par = "best")
```

The validation measure for each of these models is given below. The robust version achieves lower MAE, with all predictors retained in the ensemble (each has at least one non-zero coefficient across models).

```
R> rbind("robglm" = get_measure(best_rob), "glm" = get_measure(best_glm))
```

	nu	nummod	mae	numactive
robglm	0.004830264	20	4.389492	1000
glm	0.002894128	20	4.584595	1000

5. Conclusion

Package **spareg** can be employed for modeling high-dimensional data in a GLM framework, especially in settings where the number of predictors is much larger than the number of observations. The package provides an implementation of a computationally-efficient algorithm for sparse projected and average regression (SPAR), which combines variable screening and random projection in an ensemble of GLMs to reduce dimensionality of the predictors. Package **spareg** provides flexible classes for: i) specifying the coefficient based on which screening should be performed (either in a classical fashion, where the predictors with the largest screening coefficient are selected for subsequent analysis, or in a probabilistic fashion, where variables are sampled for inclusion with probabilities proportional to their screening coefficients), ii) generating the random projection to be employed in each marginal model, and iii) specifying the marginal models to be used in the ensemble. Screening coefficients based on marginal correlation between the predictors and the response, marginal coefficients from a GLM, and ridge coefficients are provided in the package. Moreover, several random projections are implemented: the Gaussian and sparse matrices, which are data-agnostic and satisfy the JL lemma, and the data-driven projection proposed in [Parzer *et al.* \(2025\)](#) for linear regression and extended to GLMs in [Parzer *et al.* \(2024\)](#). This data-driven approach, where information about the relationship among the responses and the predictors is incorporated in the random projection by replacing the non-zero elements of the random projection of [Clarkson and Woodruff \(2013\)](#) with the ridge coefficients obtained with a small penalty, has the advantage of ensuring that the true regression coefficients can be recovered approximately after the projection. Methodologically, the SPAR algorithm with ridge screening coefficients and data-driven random projection of [Parzer *et al.* \(2024\)](#) has been demonstrated to perform effectively in terms of predictive power and variable ranking in settings with correlated predictors and across different degrees of sparsity of the coefficient vector, making it a suitable method for applications with high-dimensional correlated predictors where the sparsity of the problem is unknown. This is the motivation behind setting the random projection in [Parzer *et al.* \(2024\)](#) as a default in **spareg**.

Moreover, the flexibility and adaptability of the **spareg** package in dealing with different types of screening, random projection and GLMs make it an attractive choice for practitioners and researchers in a wide variety of data settings. It encourages exploration of new methods for variable screening and random projections or the combination of existing approaches to tailor solutions to specific data requirements.

Computational details

The results in this paper were obtained using R 4.6.1. R itself and all packages used are available from CRAN at <https://CRAN.R-project.org/>.

```
R> sessionInfo()
```

```
R version 4.6.1 (2026-06-24)
```

```
Platform: aarch64-apple-darwin23
```

```
Running under: macOS Sonoma 14.2.1
```

```
Matrix products: default
```

```
BLAS: /Library/Frameworks/R.framework/Versions/4.6/Resources/lib/libRblas.0.dylib
```

```
LAPACK: /Library/Frameworks/R.framework/Versions/4.6/Resources/lib/libRlapack.dylib; LAPACK
```

```
locale:
```

```
[1] C/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
```

```
time zone: Europe/Vienna
```

```
tzcode source: internal
```

```
attached base packages:
```

```
[1] stats      graphics  grDevices  utils      datasets  methods
[7] base
```

```
other attached packages:
```

```
[1] spareg_1.2.0  ggplot2_4.0.3
```

```
loaded via a namespace (and not attached):
```

```
[1] generics_0.1.4      tidyr_1.3.2
[3] robustbase_0.99-7   rstatix_1.1.0
[5] shape_1.4.6.1       lattice_0.23-1
[7] magrittr_2.0.5      grid_4.6.1
[9] RColorBrewer_1.1-3  iterators_1.0.14
[11] foreach_1.5.2       glmnet_5.0
[13] Matrix_1.7-6        backports_1.5.1
[15] Formula_1.2-6       survival_3.8-9
[17] energy_1.7-12       purrr_1.2.2
[19] scales_1.4.0        codetools_0.2-20
[21] abind_1.4-8         Rdpack_2.6.6
[23] cli_3.6.6           expm_1.0-1
[25] rlang_1.3.0         rbibutils_2.4.1
[27] cowplot_1.2.0       gsl_2.1-9
[29] splines_4.6.1       withr_3.0.3
[31] VariableScreening_0.2.1 tools_4.6.1
[33] ggsignif_0.6.4      dplyr_1.2.1
[35] ggpubr_1.0.0        boot_1.3-32
```

[37] ROCR_1.0-12	broom_1.0.13
[39] vctrs_0.7.3	R6_2.6.1
[41] lifecycle_1.0.5	car_3.1-5
[43] MASS_7.3-66	pkgconfig_2.0.3
[45] gee_4.13-30	pillar_1.11.1
[47] gtable_0.3.6	glue_1.8.1
[49] Rcpp_1.1.2	DEoptimR_1.2-1
[51] tibble_3.3.1	tidyselect_1.2.1
[53] rstudioapi_0.19.0	farver_2.1.2
[55] carData_3.0-6	labeling_0.4.3
[57] compiler_4.6.1	S7_0.2.2

References

- Achlioptas D (2003). “Database-Friendly Random Projections: Johnson-Lindenstrauss with Binary Coins.” *Journal of Computer and System Sciences*, **66**(4), 671–687. ISSN 0022-0000. doi:[10.1016/s0022-0000\(03\)00025-4](https://doi.org/10.1016/s0022-0000(03)00025-4). Special Issue on PODS 2001.
- Aghila G, Siddharth R (2020). **RandPro**: *Random Projection with Classification*. doi:[10.32614/CRAN.package.randpro](https://doi.org/10.32614/CRAN.package.randpro). R package version 0.2.2.
- Ailon N, Chazelle B (2009). “The Fast Johnson–Lindenstrauss Transform and Approximate Nearest Neighbors.” *SIAM Journal on Computing*, **39**(1), 302–322. doi:[10.1137/060673096](https://doi.org/10.1137/060673096).
- Cannings TI, Samworth RJ (2017). “Random-Projection Ensemble Classification.” *Journal of the Royal Statistical Society B*, **79**(4), 959–1035. doi:[10.1111/rssb.12228](https://doi.org/10.1111/rssb.12228).
- Cannings TI, Samworth RJ (2021). **RPEnsemble**: *Random Projection Ensemble Classification*. doi:[10.32614/CRAN.package.rpensemble](https://doi.org/10.32614/CRAN.package.rpensemble). R package version 0.5.
- Cantoni E, Ronchetti E (2001). “Robust Inference for Generalized Linear Models.” *Journal of the American Statistical Association*, **96**(455), 1022–1030. doi:[10.1198/016214501753209004](https://doi.org/10.1198/016214501753209004).
- Cheng X, Wang H, Zhu L, Zhong W, Zhou H (2024). **MFSIS**: *Model-Free Sure Independent Screening Procedures*. doi:[10.32614/CRAN.package.mfsis](https://doi.org/10.32614/CRAN.package.mfsis). R package version 0.2.1.
- Cho H, Fryzlewicz P (2012). “High Dimensional Variable Selection Via Tilting.” *Journal of the Royal Statistical Society B*, **74**(3), 593–622. doi:[10.1111/j.1467-9868.2011.01023.x](https://doi.org/10.1111/j.1467-9868.2011.01023.x).
- Clarkson KL, Woodruff DP (2013). “Low Rank Approximation and Regression in Input Sparsity Time.” In *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing*, STOC ’13, pp. 81–90. Association for Computing Machinery, New York, NY, USA. ISBN 9781450320290. doi:[10.1145/2488608.2488620](https://doi.org/10.1145/2488608.2488620).
- Cui H, Li R, Zhong W (2015). “Model-Free Feature Screening for Ultrahigh Dimensional Discriminant Analysis.” *Journal of the American Statistical Association*, **110**(510), 630–641. doi:[10.1080/01621459.2014.920256](https://doi.org/10.1080/01621459.2014.920256).

- Fan J, Feng Y, Song R (2011). “Nonparametric Independence Screening in Sparse Ultra-High-Dimensional Additive Models.” *Journal of the American Statistical Association*, **106**(494), 544–557. doi:[10.1198/jasa.2011.tm09779](https://doi.org/10.1198/jasa.2011.tm09779).
- Fan J, Feng Y, Wu Y (2010). “High-Dimensional Variable Selection for Cox’s Proportional Hazards Model.” In *Borrowing Strength: Theory Powering Applications—a Festschrift for Lawrence D. Brown*, volume 6, pp. 70–87. Institute of Mathematical Statistics. doi:[10.1214/10-imscol1606](https://doi.org/10.1214/10-imscol1606).
- Fan J, Lv J (2007). “Sure Independence Screening for Ultra-High Dimensional Feature Space.” *Journal of the Royal Statistical Society B*, **70**. doi:[10.1111/j.1467-9868.2008.00674.x](https://doi.org/10.1111/j.1467-9868.2008.00674.x).
- Fan J, Samworth R, Wu Y (2009). “Ultrahigh Dimensional Feature Selection: Beyond the Linear Model.” *Journal of Machine Learning Research*, **10**, 2013–2038. URL <https://jmlr.csail.mit.edu/papers/v10/fan09a.html>.
- Fan J, Song R (2010). “Sure Independence Screening in Generalized Linear Models with NP-Dimensionality.” *The Annals of Statistics*, **38**(6), 3567 – 3604. doi:[10.1214/10-aos798](https://doi.org/10.1214/10-aos798).
- Frankl P, Maehara H (1988). “The Johnson-Lindenstrauss Lemma and the Sphericity of Some Graphs.” *Journal of Combinatorial Theory, Series B*, **44**(3), 355–362. ISSN 0095-8956. doi:[10.1016/0095-8956\(88\)90043-3](https://doi.org/10.1016/0095-8956(88)90043-3).
- Friedman J, Hastie T, Tibshirani R (2010). “Regularization Paths for Generalized Linear Models via Coordinate Descent.” *Journal of Statistical Software*, **33**(1), 1–22. doi:[10.18637/jss.v033.i01](https://doi.org/10.18637/jss.v033.i01).
- Gataric M, Wang T, Samworth RJ (2019). **SPCAvRP**: Sparse Principal Component Analysis via Random Projections (SPCAvRP). doi:[10.32614/CRAN.package.spcavrp](https://doi.org/10.32614/CRAN.package.spcavrp). R package version 0.4.
- Gaujoux R (2025). **doRNG**: Generic Reproducible Parallel Backend for **foreach** Loops. doi:[10.32614/CRAN.package.doRNG](https://doi.org/10.32614/CRAN.package.doRNG). R package version 1.8.6.2.
- Guedj B, Desikan BS (2018). “**pycobra**: A Python Toolbox for Ensemble Learning and Visualisation.” *Journal of Machine Learning Research*, **18**(190), 1–5. URL <http://jmlr.org/papers/v18/17-228.html>.
- H2Oai (2016). **H2O**. H2O version 3.10.0.8, URL <https://github.com/h2oai/h2o-3>.
- Johnson W, Lindenstrauss J (1984). “Extensions of Lipschitz Maps into a Hilbert Space.” *Contemporary Mathematics*, **26**, 189–206. doi:[10.1090/conm/026/737400](https://doi.org/10.1090/conm/026/737400).
- Ke C (2023). “Sufficient Variable Screening With High-Dimensional Controls.” *Electronic Journal of Statistics*, **17**(2), 2139 – 2179. doi:[10.1214/23-ejs2150](https://doi.org/10.1214/23-ejs2150).
- Li P, Hastie TJ, Church KW (2006). “Very Sparse Random Projections.” In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD ’06*, pp. 287–296. Association for Computing Machinery, New York, NY, USA. ISBN 1595933395. doi:[10.1145/1150402.1150436](https://doi.org/10.1145/1150402.1150436).

- Li R, Huang L, Dziak J (2022). **VariableScreening**: *High-Dimensional Screening for Semi-parametric Longitudinal Regression*. doi:10.32614/CRAN.package.variablescreening. R package version 0.2.1.
- Li R, Zhong W, Zhu L (2012). “Feature Screening via Distance Correlation Learning.” *Journal of the American Statistical Association*, **107**(499), 1129–1139. doi:10.1080/01621459.2012.695654.
- Liu J, Liu W, Ke Y, Li R (2022). “Model-Free Feature Screening and FDR Control With Knockoff Features.” *Journal of the American Statistical Association*, **117**(537), 428–443. doi:10.1080/01621459.2020.1783274.
- Ma X, Zhang J (2016). “Robust Model-Free Feature Screening via Quantile Correlation.” *Journal of Multivariate Analysis*, **143**, 472–480. doi:10.1016/j.jmva.2015.10.010.
- Mai Q, Zou H (2013). “The Kolmogorov Filter for Variable Screening in High-Dimensional Binary Classification.” *Biometrika*, **100**(1), 229–234. doi:10.1093/biomet/ass062.
- Mai Q, Zou H (2015). “The Fused Kolmogorov Filter: A Nonparametric Model-Free Screening Method.” *The Annals of Statistics*, **43**(4), 1471 – 1497. doi:10.1214/14-aos1303.
- Matoušek J (2008). “On Variants of the Johnson–Lindenstrauss Lemma.” *Random Structures & Algorithms*, **33**(2), 142–156. doi:10.1002/rsa.20218.
- Microsoft, Weston S (2022). **foreach**: *Provides Foreach Looping Construct*. doi:10.32614/CRAN.package.foreach. R package version 1.5.2.
- Microsoft Corporation, Weston S (2022). **doParallel**: *Foreach Parallel Adaptor for the parallel Package*. doi:10.32614/CRAN.package.doparallel. R package version 1.0.17.
- Mukhopadhyay M, Dunson DB (2020). “Targeted Random Projection for Prediction From High-Dimensional Features.” *Journal of the American Statistical Association*, **115**(532), 1998–2010. doi:10.1080/01621459.2019.1677240.
- Parzer R, Filzmoser P, Vana-Gür L (2024). “Data-Driven Random Projection and Screening for High-Dimensional Generalized Linear Models.” *Technical Report 2410.00971*, arXiv.org E-Print Archive. doi:10.48550/arxiv.2410.00971.
- Parzer R, Filzmoser P, Vana-Gür L (2025). “Sparse Data-Driven Random Projection in Regression for High-Dimensional Data.” *Journal of Data Science, Statistics, and Visualisation*, **5**(5). doi:10.52933/jdssv.v5i5.138.
- Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, et al. (2011). “**scikit-Learn**: Machine Learning in Python.” *Journal of Machine Learning Research*, **12**(Oct), 2825–2830. URL <https://www.jmlr.org/papers/v12/pedregosa11a.html>.
- R Core Team (2025). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL <https://www.R-project.org/>.
- Ryder N, Karnin Z, Liberty E (2019). “Asymmetric Random Projections.” *Technical Report 1906.09489*, arXiv.org E-Print Archive. doi:10.48550/arxiv.1906.09489.

- Saldana DF, Feng Y (2018). “**SIS**: An R Package for Sure Independence Screening in Ultrahigh-Dimensional Statistical Models.” *Journal of Statistical Software*, **83**(2), 1–25. doi:10.18637/jss.v083.i02.
- Siddharth R, Aghila G (2020). “**RandPro** – A Practical Implementation of Random Projection-Based Feature Extraction for High Dimensional Multivariate Data Analysis in R.” *SoftwareX*, **12**, 100629. ISSN 2352-7110. doi:10.1016/j.softx.2020.100629.
- Song L, Langfelder P (2022). **randomGLM**: Random General Linear Model Prediction. doi:10.32614/CRAN.package.randomGLM. R package version 1.10-1.
- StataCorp (2025). *Stata Statistical Software: Release 19*. StataCorp LLC, College Station, TX. URL <https://www.stata.com/>.
- Tay JK, Narasimhan B, Hastie T (2023). “Elastic Net Regularization Paths for All Generalized Linear Models.” *Journal of Statistical Software*, **106**(1), 1–31. doi:10.18637/jss.v106.i01.
- Thanei GA, Heinze C, Meinshausen N (2017). *Random Projections for Large-Scale Regression*, pp. 51–68. Springer International Publishing. doi:10.1007/978-3-319-41573-4_3.
- Tian Y, Feng Y (2021). **RaSEn**: Random Subspace Ensemble Classification and Variable Screening. doi:10.32614/CRAN.package.rasen. R package version 3.0.0.
- Todorov V, Filzmoser P (2009). “An Object-Oriented Framework for Robust Multivariate Analysis.” *Journal of Statistical Software*, **32**(3), 1–47. doi:10.18637/jss.v032.i03.
- Van Rossum G, et al. (2011). *Python Programming Language*. URL <https://www.python.org>.
- Vana-Gür L, Parzer R, Filzmoser P (2025). **spareg**: Sparse Projected Averaged Regression in R. doi:10.32614/CRAN.package.spareg. R package version 1.1.1.
- Wang X, Leng C (2016). “High-Dimensional Ordinary Least-Squares Projection for Screening Variables.” *Journal of the Royal Statistical Society B*, **78**, 589–611. doi:10.1111/rssb.12127.
- Wang X, Pan W, Hu W, Tian Y, Zhang H (2015). “Conditional Distance Correlation.” *Journal of the American Statistical Association*, **110**(512), 1726–1734. doi:10.1080/01621459.2014.993081.
- Wickham H (2016). **ggplot2**: *Elegant Graphics for Data Analysis*. Springer-Verlag. ISBN 978-3-319-24277-4. doi:10.1007/978-0-387-98141-3.
- Xu C, Chen J (2014). “The Sparse MLE for Ultrahigh-Dimensional Feature Screening.” *Journal of the American Statistical Association*, **109**(507), 1257–1269. doi:10.1080/01621459.2013.879531.
- Zhu LP, Li L, Li R, Zhu LX (2011). “Model-Free Feature Screening for Ultrahigh-Dimensional Data.” *Journal of the American Statistical Association*, **106**(496), 1464–1475. doi:10.1198/jasa.2011.tm10563.

Affiliation:

Laura Vana-Gür

Computational Statistics (CSTAT)

Institute of Statistics and Mathematical Methods in Economics

TU Wien

Wiedner Hauptstraße 8-10

1040 Vienna, Austria

E-mail: laura.vana.guer@tuwien.ac.at

Roman Parzer

Computational Statistics (CSTAT)

Institute of Statistics and Mathematical Methods in Economics

TU Wien

Wiedner Hauptstraße 8-10

1040 Vienna, Austria

E-mail: romanparzer1@gmail.com

Peter Filzmoser

Computational Statistics (CSTAT)

Institute of Statistics and Mathematical Methods in Economics

TU Wien

Wiedner Hauptstraße 8-10

1040 Vienna, Austria

E-mail: peter.filzmoser@tuwien.ac.at