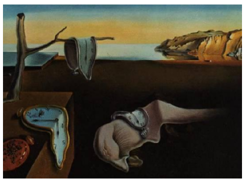


Journal of Statistical Software

MMMMMM YYYY, Volume VV, Issue II. *doi: 10.18637/jss.v000.i00*

SPAR: Sparse Projected Averaged Regression in R

Roman Parzer
TU Wien

Peter Filzmoser
TU Wien

Laura Vana Gür
TU Wien

Abstract

SPAR is a package for building predictive generalized linear models (GLMs) with high-dimensional (HD) predictors in R. In package **SPAR**, probabilistic variable screening and random projection of the predictors are performed to obtain an ensemble of GLMs, which are then averaged to obtain predictions in an high-dimensional regression setting.

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

1. Introduction

SPAR is a package for building predictive generalized linear models (GLMs) with high-dimensional (HD) predictors in R. In package **SPAR**, probabilistic variable screening and random projection of the predictors are performed to obtain an ensemble of GLMs, which are then averaged to obtain predictions in an high-dimensional regression setting.

Random projection is a computationally-efficient method which linearly maps a set of points in high dimensions into a much lower-dimensional space while approximately preserving pairwise distances. For very large p, random projection can suffer from overfitting, as too many irrelevant predictors are being considered for prediction purposes (Mukhopadhyay and Dunson 2020). Therefore, screening out irrelevant variables before performing the random projection is advisable in order to tackle this issue. The screening can be performed in a probabilistic fashion, by randomly sampling covariates for inclusion in the model based on probabilities proportional to an importance measure (as opposed to random subspace sampling employed in e.g., random forests). Finally, in practice the information from multiple such screening and projections can be combined by averaging, in order to reduce the variance introduced by the random sampling (of both projections and screening indicators) (Thanei, Heinze, and Meinshausen 2017).

Several packages which provide functionality for random projections are available for R. Package **RandPro** (Aghila and Siddharth 2020; Siddharth and Aghila 2020) allows for four different random projection matrices to be applied to the predictor matrix before employing one of k~nearest neighbor, support vector machine or naive Bayes classifier. 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. Package **RPEnsembleR** (Cannings and Samworth 2021) implements the same idea of "carefully-selected" random projections when building an ensemble of classifiers. For Python van Rossum *et al.* (2011) the **sklearn.random_projection** module implements two types of unstructured random matrix, namely Gaussian random matrix and sparse random matrix.

On the other hand, there are a multitude of packages dealing with variable screening on the Comprehensive R Archive Network (CRAN). The (iterative) sure independence screening procedure and extensions in Fan and Lv (2007), Fan and Song (2010), Fan, Feng, and Wu (2010) are implemented in package **SIS** (Saldana and Feng 2018), which also provides functionality for estimating a penalized generalized linear model or a cox regression model for the variables picked by the screening procedure.

Package **VariableScreening** (Li, Huang, and Dziak 2022) implements screening for iid data, varying-coefficient models, and longitudinal data using different screening methods: Sure Independent Ranking and Screening – which ranks the predictors by their correlation with the rank-ordered response (SIRS), Distance Correlation Sure Independence Screening – a non-parametric extension of the correlation coefficient (DC-SIS), MV Sure Independence Screening – using the mean conditional variance measure (MV-SIS).

A collection of model-free screening techniques such as SIRS, DC-SIS, MV-SIS, the fused Kolmogorov filter (Mai and Zou 2015), the projection correlation method using knock-off features (Wanjun Liu and Li 2022), are provided in package **MFSIS** (Cheng, Wang, Zhu, Zhong, and Zhou 2024). Package **tilting** (Cho and Fryzlewicz 2016) implements an algorithm for variable selection in high-dimensional linear regression using tilted correlation, which takes into account high correlations among the variables in a data-driven way. Feature screening based on conditional distance correlation (Wang, Pan, Hu, Tian, and Zhang 2015) can be performed with the **cdcsis** package (Hu, Huang, Pan, Wang, Wen, Tian, Zhang, and Zhu 2024) while package **QCSIS** (Ma, Zhang, and Zhou 2015) implements screening based on (composite) quantile correlation.

Package **LqG** (Wu, Li, and Li 2022) provides a group screening procedure that is based on maximum Lq-likelihood estimation, to simultaneously account for the group structure and data contamination in variable screening.

Feature screening using an $L1$ fusion penalty can be performed with package **fusionclust** (Banerjee, Mukherjee, and Radchenko 2017). Package **SMLE** (Zang, Xu, and Burkett 2020) implements joint feature screening via sparse MLE (Xu and Chen 2014) in high-dimensional linear, logistic, and Poisson models. Package **TSGSIS** (Fang, Wang, and Hsiung 2017b) provides a high-dimensional grouped variable selection approach for detecting interactions that may not have marginal effects in high dimensional linear and logistic regression (Fang, Wang, and Hsiung 2017a).

Package **RaSEn** (Tian and Feng 2021) implements the RaSE algorithm for ensemble classification and classification problems, where random subspaces are generated and the optimal one is chosen to train a weak learner on the basis of some criterion. Various choices of base classifiers are implemented, for instance, linear discriminant analysis, quadratic discriminant analysis, k-nearest neighbor, logistic or linear regression, decision trees, random forest, support vector machines. The selected percentages of variables can be employed for variable screening.

Package **Ball** (Zhu, Pan, Zheng, and Wang 2021) provides functionality for variable screening using ball statistics, which is appropriate for shape, directional, compositional and symmetric positive definite matrix data.

Package **BayesS5** (Shin and Tian 2020) implements Bayesian variable selection using simplified shotgun stochastic search algorithm with screening (Shin, Bhattacharya, and Johnson 2017) while package **bravo** (Li, Chakraborty, Dutta, and Roy 2024) implements the Bayesian iterative screening method proposed in (Wang, Dutta, and Roy 2021).

The rest of the paper is organized as follows: Section 2 provides the methodological details of the implemented algorithm. The package is described in Section 3. Section 4 contains two examples of employing the package on real data sets. Finally, Section 5 concludes.

2. Methods

2.1. Variable screening

The general idea of variable screening is to select a (small) subset of variables, based on some marginal utility measure for the predictors, and disregard the rest for further analysis. In their seminal work on sure independence screening (SIS), Fan and Lv (2007) propose to use the vector of marginal empirical correlations $\hat{\alpha} = (\alpha_1, \dots, \alpha_p)' \in \mathbb{R}^p, \alpha_j = \mathrm{Cor}(X_{.j}, y)$ for variable screening in a linear regression setting by selecting the variable set $\mathcal{A}_\gamma = \{j \in [p] : |w_j| > \gamma\}$ depending on a threshold $\gamma > 0$, where $[p] = \{1, \dots, p\}$. Under certain technical conditions, where p grows exponentially with n, they show that this procedure has the *sure screening property*

$$\mathbb{P}(\mathcal{A} \subset \mathcal{A}_{\gamma_n}) \to 1 \text{ for } n \to \infty$$

with an explicit exponential rate of convergence, where $\mathcal{A} = \{j \in [p] : \beta_j \neq 0\}$ is the set of truly active variables. These conditions imply that $\mathcal{A}$ and $\mathcal{A}_{\gamma_n}$ contain less than n variables. One of the critical conditions is that on the population level for some fixed $i \in [n]$, $\min_{j \in \mathcal{A}} |\mathrm{Cov}(y_i/\beta_j, x_{ij})| \geq c$ for some constant $c > 0$, which rules out practically possible scenarios where an important variable is marginally uncorrelated to the response. Fan and Song (2010) extend the approach to GLMs, where the screening is performed based on the log-likelihood of the GLM containing only X_j as a predictor: $\hat{\alpha}_j =: \min_{\beta_j \in \mathbb{R}} \sum_{i=1}^{n} -\ell(\beta; y_i, x_{ij})$.

A rich stream of literature focuses on developing semi- or non-parametric alternatives to SIS which should be more robust to model misspecification. For linear regression, approaches include using the ranked correlation (Zhu, Li, Li, and Zhu 2011), (conditional) distance correlation (Li, Zhong, and Zhu 2012, @wang2015conditional). 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), using the mean conditional variance, i.e., the expectation in X_j of the variance in the response of the conditional cumulative distribution function $\mathbb{P}(X \leq x|Y)$ (Cui, Li, and Zhong 2015). Ke (2023) propose 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.

To account for multicollinearity among the predictors, which can cause relevant predictors to be marginally uncorrelated with the response, various extensions have been proposed. In a linear regression setting, Wang and Leng (2016) propose employing the ridge estimator when the penalty term converges to zero while Cho and Fryzlewicz (2012) propose using the tilted correlation, i.e., the correlation of a tilted version of X_j with y. For discrete outcomes, joint feature screening Xu and Chen (2014) has been proposed.

2.2. Random projection

The random projection method relies on the Johnson-Lindenstrauss (JL) lemma (Johnson and Lindenstrauss 1984), which asserts that there exists a random map $\Phi \in \mathbb{R}^{m\times p}$ that embeds any set of points in p-dimensional Euclidean space collected in the rows of $X \in \mathbb{R}^{n\times p}$ into a m-dimensional Euclidean space with $m < \mathcal{O}(\log n/\varepsilon^2)$ so that all pairwise distances are maintained within a factor of $1 \pm \varepsilon$, for any $0 < \varepsilon < 1$.

The random map Φ should be chosen such that it fulfills certain conditions (see Johnson and Lindenstrauss 1984). 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). Common choices are:

- Normal distribution.: $\Phi_{ij} \overset{iid}{\sim} N(0,1)$ (Frankl and Maehara 1988) or $\Phi_{ij} = \begin{cases} \sim N(0, 1/\sqrt{\psi}) & \text{with prob. } \psi \\ 0 & \text{with prob. } 1-\psi \end{cases}$ (Matoušek 2008),
- Rademacher distribution (Achlioptas 2003; Li, Hastie, and Church 2006)

  $$\Phi_{ij} = \begin{cases} \pm 1/\sqrt{\psi} & \text{with prob. } \psi/2 \\ 0 & \text{with prob. } 1-\psi, \quad 0 < \psi \leq 1, \end{cases},$$

  where Achlioptas (2003) shows results for $\psi = 1$ and $\psi = 1/3$ while Li *et al.* (2006) recommend using $\psi = 1/\sqrt{p}$ to obtain very sparse matrices.

Distributions which are not sub-Gaussian, such as standard Cauchy, have also been proposed in the literature to tackle scenarios where the data is high-dimensional, non-sparse, and heavy-tailed by preserving approximate ℓ_1 distances (see e.g., Li, Hastie, and Church 2007).

An orthonormalization is usually applied $(\Phi\Phi^\top)^{-1/2}\Phi$. Orthonormalization can constitute a computational bottleneck for the random projection method, however, in high-dimensions it can be omitted.

To speed computations, Ailon and Chazelle (2009) propose the fast Johnson- Lindenstrauss transform (FJLT), where the random projection matrix is given by $\Phi = PHD$ with P random and sparse, $P_{ij} \sim N(0, 1/q)$ with probability $1/q$ and 0 otherwise, H the normalized Hadamard (orthogonal) matrix $H_{ij} = p^{-1/2}(-1)^{\langle i-1, j-1\rangle}$, where $\langle i, j\rangle$ is the dot-product of the m-bit vectors i, j expressed in binary, and $D = \text{diag}(\pm 1)$ is a diagonal matrix with random elements D_{ii}.

Clarkson and Woodruff (2013) propose a sparse embedding matrix $\Phi = BD$, where $B \in \{0, 1\}^{m\times p}$ is random binary matrix and D is a $p \times p$ diagonal matrix with $(D_{ii} + 1)/2 \sim \text{Unif}\{0, 1\}$, and prove that the dimension m is bounded by a polynomial in $r\varepsilon^{-1}$ for $0 < \varepsilon < 1$ and r being the rank of X. While this is generally larger than that of FJLT, the sparse embedding matrix requires less time to compute ΦX compared to other subspace embeddings.

Parzer, Filzmoser, and Vana-Gür (2024) propose employing $D_{ii} = \hat{\alpha}$ in the sparse embedding matrix of Clarkson and Woodruff (2013), $\hat{\alpha}$ is a screening coefficient in the regression such as the ridge or the HOLP coefficients, and show that the proposed projection increases the predictive performance in a linear regression setting.

2.3. Algorithm

- choose family with corresponding log-likelihood $\ell(.)$ and link
- standardize predictors $X : n \times p$
- calculate screening coefficients $\hat{\alpha}$ e.g.,
  - ridge: $\hat{\alpha} =: \text{argmin}_{\beta\in\mathbb{R}^p} \sum_{i=1}^n -\ell(\beta; y_i, x_i) + \frac{\varepsilon}{2}\sum_{j=1}^p \beta_j^2$, $\varepsilon > 0$
  - marginal likelihood: $\hat{\alpha}_j =: \min_{\beta_j\in\mathbb{R}} \sum_{i=1}^n -\ell(\beta; y_i, x_{ij})$
- For $k = 1, \dots, M$ models:
  - draw $2n$ predictors with probabilities $p_j \propto |\hat{\alpha}_j|$ yielding screening index set $I_k = \{j_1^k, \dots, j_{2n}^k\} \subset [p]$
  - project remaining variables to dim. $m_k \sim \text{Unif}\{\log(p), \dots, n/2\}$ using **projection matrix** Φ_k to obtain $Z_k = X_{\cdot I_k}\Phi_k^\top \in \mathbb{R}^{n\times m_k}$:
  - fit a **GLM** of y against Z_k (with small L_2-penalty Tay, Narasimhan, and Hastie (2023)) to obtain estimated coefficients $\gamma^k \in \mathbb{R}^{m_k}$ and $\hat{\beta}_{I_k}^k = \Phi_k'\gamma^k$, $\hat{\beta}_{\bar{I}_k}^k = 0$.
- for a given threshold $\lambda > 0$, set all $\hat{\beta}_j^k$ with $|\hat{\beta}_j^k| < \lambda$ to 0 for all j, k
- *Optional:* choose M and λ via cross-validation by repeating steps 1 to 4 for each fold and evaluating a prediction performance measure on the withheld fold; and choose

$$(M_{\text{best}}, \lambda_{\text{best}}) = \text{argmin}_{M,\lambda}\text{Dev}(M, \lambda) \tag{1}$$

- combine via **simple average** $\hat{\beta} = \sum_{k=1}^{M} \hat{\beta}^k / M$
- output the estimated coefficients and predictions for the chosen M and λ

3. Software

The two main functions are:

```
spar(x, y, family = gaussian(), ...)
```

and

```
spar.cv(x, y, family = gaussian(), nfolds = 10L, ...)
```

Most important arguments:

- `x` $n \times p$ numeric matrix of predictor variables.
- `y` numeric response vector of length n.
- `family` object from `stats::family()`.
- `type.rpm` type of random projection matrix to be employed; one of `"cwdatadriven"`, `"cw"` Clarkson and Woodruff (2013), `"gaussian"`, `"sparse"`.
- `type.screening` measure by which the coefficients are screened; `"ridge"` performs screening based on ridge regression, `"marglik"` marginal likelihood of fitting a GLM for each predictor, `"corr"` correlation of the response with each predictor.
- `type.measure` loss to use for choosing λ and M; defaults to `"deviance"` (available for all families). Other options are `"mse"` or `"mae"` (for all families), `"class"` and `"1-auc"` for `"binomial"`.
- `nlambda` number of λ values to be considered for thresholding and optionally `lambdas`, a vector of values.
- `nummods` vector containing the size of the different ensembles M to consider for the prediction.

Methods `print`, `plot`, `coef`, `predict` are available.

Name	Random projection method
`gaussian`	Standard Gaussian
`sparse`	Rademacher

Name	Random projection method
`cw`	sparse embedding matrix
`cwdatadriven`	data driven sparse embedding matrix

Table 1: Overview of implemented random projection matrices.

4. Illustrations

4.1. Face image data

We illustrate the package on a data set containing $n = 698$ black and white face images of size $p = 64 \times 64 = 4096$ and the faces' horizontal looking direction angle as the response variable. The Isomap face data can be found online on https://web.archive.org/web/20160913051505/http://isomap. stanford.edu/datasets.html

```
library("R.matlab")
```

```
R.matlab v3.7.0 (2022-08-25 21:52:34 UTC) successfully loaded. See ?R.matlab for help.



Attaching package: 'R.matlab'


The following objects are masked from 'package:base':

    getOption, isOpen
```

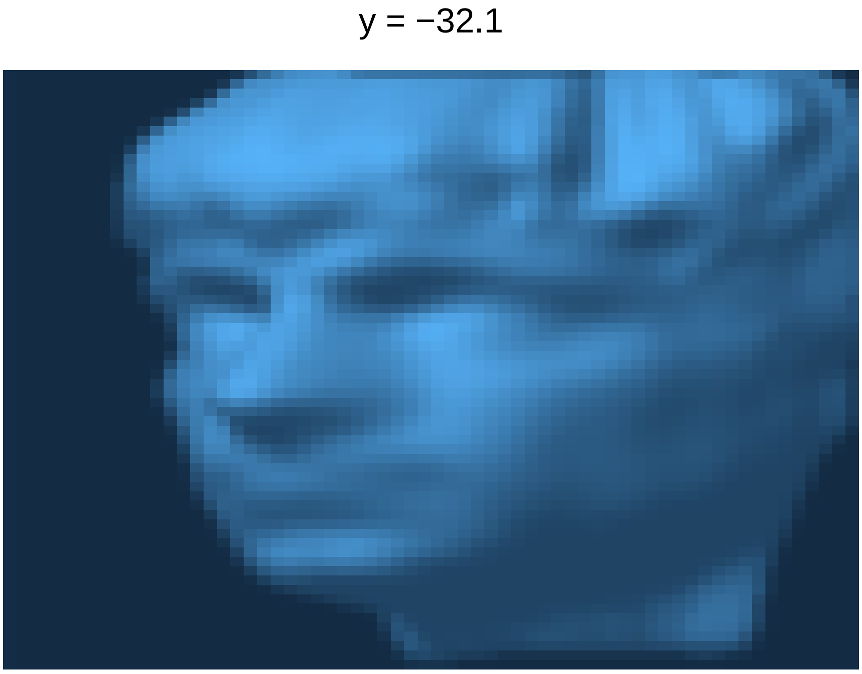
```
temp <- tempdir()
download.file("https://web.archive.org/web/20150922051706/http://isomap.stanford.edu/face_
system(sprintf('uncompress %s', paste0(temp, "/face_data.mat.Z")))
facedata <- readMat(file.path(temp, "face_data.mat"))

x <- t(facedata$images)
y <- facedata$poses[1,]
```

We can visualize e.g., the first observation in this data set by:

```
library(ggplot2)
```

```
Warning: package 'ggplot2' was built under R version 4.2.3
```

```
i <- 1
ggplot(data.frame(X = rep(1:64,each=64),Y = rep(64:1,64),
                  Z = facedata$images[,i]),
       aes(X, Y, fill = Z)) +
  geom_tile() +
  theme_void() +
  ggtitle(paste0("y = ",round(facedata$poses[1,i],1))) +
  theme(legend.position = "none",
        plot.title = element_text(hjust = 0.5))
```



We can split the data into training vs test sample:

```
set.seed(1234)
ntot <- length(y)
ntest <- ntot * 0.25
testind <- sample(1:ntot, ntest, replace=FALSE)
xtrain <- as.matrix(x[-testind, ])
const_col_ind <- which(apply(xtrain,2,sd)<0.01)
if (length(const_col_ind)>0) {
  xtrain <- xtrain[,-const_col_ind]
}
ytrain <- y[-testind]
xtest <- as.matrix(x[testind, -const_col_ind])
ytest <- y[testind]
```

We can now estimate the model on the training data:

```
library(SPAR)
spar_faces <- spar.cv(xtrain, ytrain,
                      family = gaussian("identity"),
                      nummods = c(5, 10, 20, 50),
                      type.measure = "mse")
```

```
Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge
```

```
Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge
```

```
Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge
```

```
Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge
```

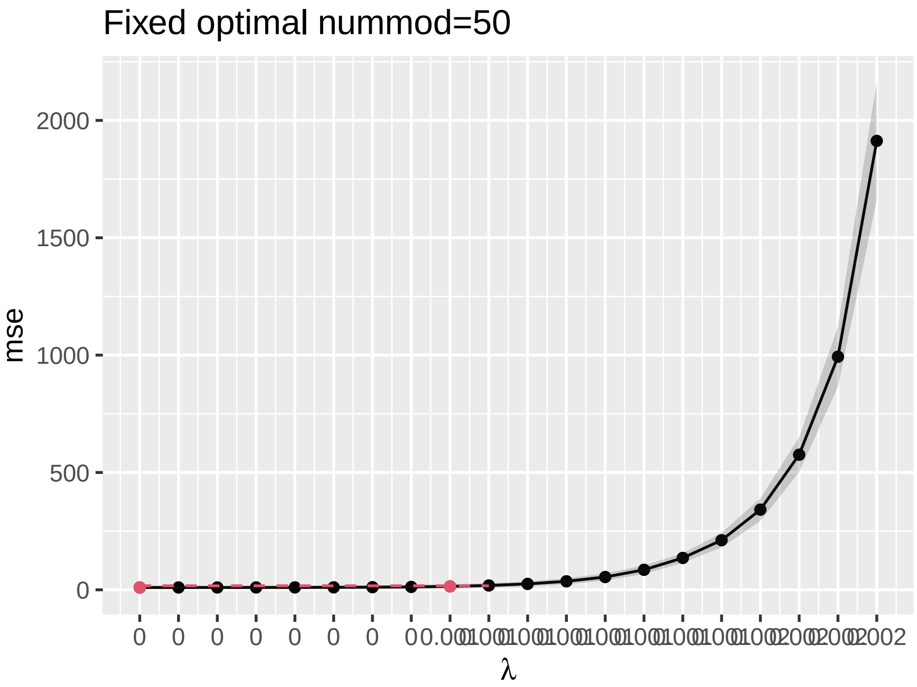
```
Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

Warning: glmnet.fit: algorithm did not converge

spar_faces
```

```
SPAR.cv object:
Smallest CV-Meas 10.0 reached for nummod=50, lambda=0.000 leading to 3682 / 3833 active pr
Summary of those non-zero coefficients:
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-33.85315  -0.14501   0.00034   0.01676   0.17048  36.39650

Sparsest coefficient within one standard error of best CV-Meas reached for nummod=5, lambd
leading to 1684 / 3833 active predictors with CV-Meas 14.6.
Summary of those non-zero coefficients:
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-34.63019  -0.51653   0.04657   0.02366   0.58570  42.34801
```

```
# undebug(spar)
#
# eig <- eigen(tcrossprod(z),symmetric = TRUE)
# myinv <- tcrossprod(eig$vectors[,eig$values>1e-8]%*%diag(1/sqrt(eig$values[eig$values>1e
# HOLP <- as.numeric(crossprod(z,myinv%*%yz))
#
# n <- NROW(z)
# p <- NCOL(z)
# tmp_sc <- apply(z,2,function(col)sqrt(var(col)*(n-1)/n))
# z2 <- scale(z,center=colMeans(z),scale=tmp_sc)
# lam_max <- 1000 * max(abs(t(yz)%*%z2[,tmp_sc>0]))/n*family$mu.eta(family$linkfun(mean(yz
# dev.ratio_cutoff <- 0.999
# glmnet_res <- glmnet::glmnet(x=z, y=yz, family = family, alpha=0,lambda.min.ratio = min(
# lam <- min(glmnet_res$lambda[glmnet_res$dev.ratio<=dev.ratio_cutoff])
# scr_coef <- coef(glmnet_res,s=lam)[-1]
# scr_coef
#
# apply(glmnet_res$beta,2,function(tmpcoef)cor(tmpcoef,HOLP))
#
# cor(scr_coef,HOLP)
```

The `plot` method for '`spar.cv`' objects displays by default the measure employed in the cross validation (in this case MSE) for a grid of λ values, where the number of models is fixed to the value found to perform best in cross-validation exercise:

```
plot(spar_faces)
```



```
# plot(spar_faces,plot_type = "Val_numAct")
```

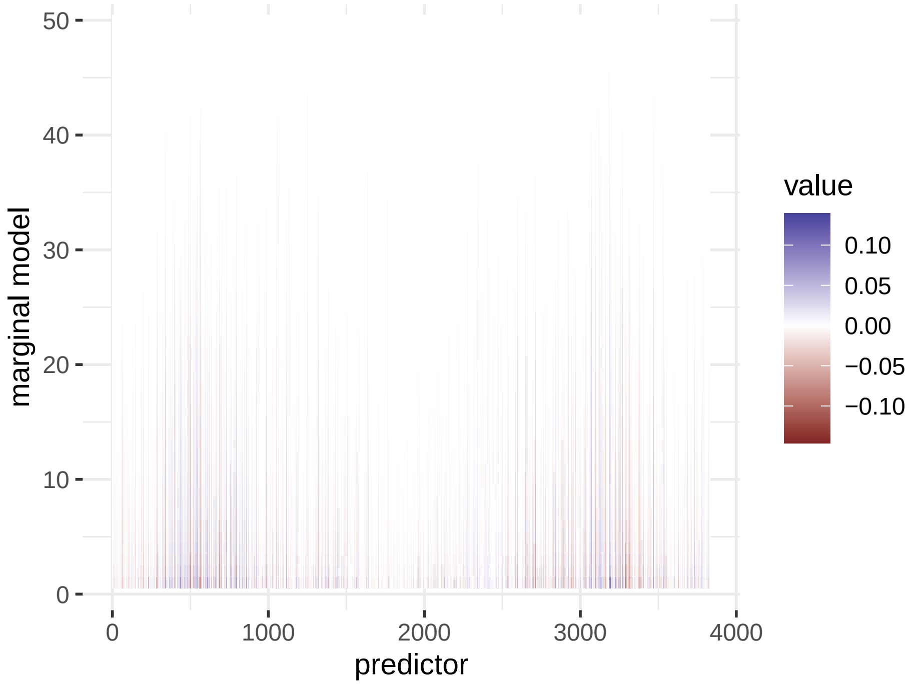
The coefficients of the different variables (in this example pixels) obtained by averaging over the coefficients the marginal models (for optimal λ and number of models) are given by:

```
face_coef <- coef(spar_faces, opt_par = "best")
str(face_coef)
```

```
List of 4
 $ intercept: num -14.5
 $ beta     : num [1:3833] -0.63 -0.1982 0 -0.0989 1.0564 ...
 $ nummod   : num 50
 $ lambda   : num 0
```

The coefficients from each of the marginal models (before averaging) can be plotted using the `plot(..., plot_type = "coefs")`

```
plot(spar_faces, "coefs")
```



The `predict()` function can be applied to the ‘`spar.cv`’ object:

```
ynew <- predict(spar_faces, xnew = xtest)
```

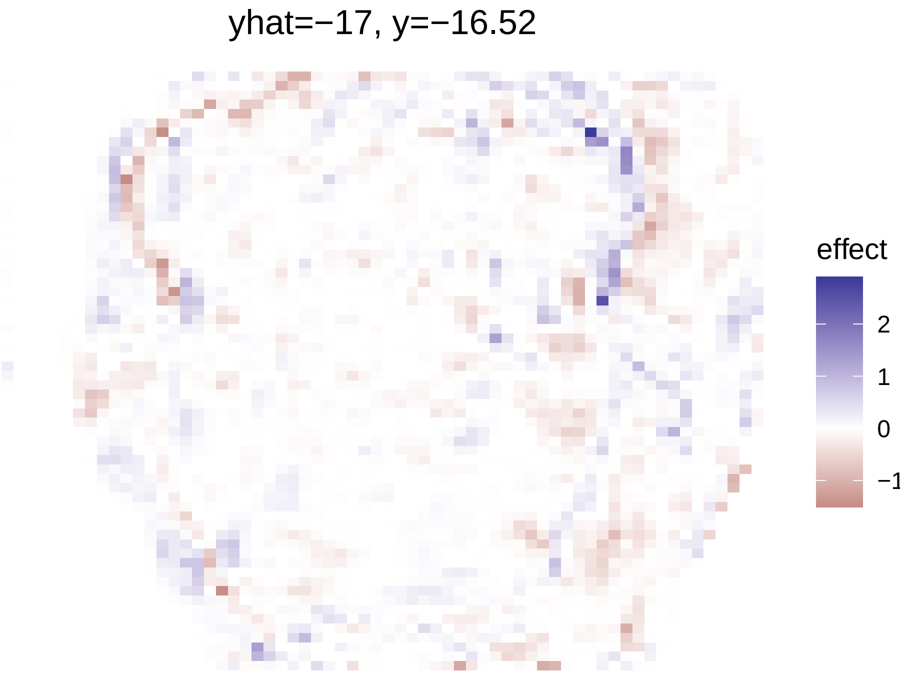
In the high-dimensional setting it is interesting to look at the relative mean square prediction error which compares the MSE to the MSE of a model containing only an intercept:

```
rMSPEconst <- mean((ytest - mean(y))^2)
mean((ynew-ytest)^2)/rMSPEconst
```

```
[1] 0.01315477
```

Additionally, for this data set, one can visualize the effect of each pixel $\hat{\beta}_j x^{\mathrm{new}}_{i,j}$ in predicting the face orientation in a given image e.g., 9th in the test set:

```
i <- 9
shap_vals <- numeric(64^2)
shap_vals[-const_col_ind] <- xtest[i,] * face_coef$beta
plot4 <- ggplot(data.frame(X = rep(1:64, each = 64),
                           Y = rep(64:1, 64),
                           effect = shap_vals),
                aes(X, Y, fill= effect)) +
  geom_tile() +
  theme_void() +
  scale_fill_gradient2() +
  ggtitle(paste0("yhat=",round(ynew[i],2),", y=",round(ytest[i],2))) +
  theme(plot.title = element_text(hjust = 0.5))
plot4
```



```
# y[testind[i]]
# sum(xtest[i,] * face_coef$beta) + face_coef$intercept
```

4.2. Darwin data set

The Darwin dataset (Cilia, De Gregorio, De Stefano, Fontanella, Marcelli, and Parziale 2022) contains a binary response for Alzheimer's disease (AD) together with extracted features from 25 handwriting tests (18 features per task) for 89 AD patients and 85 healthy people ($n = 174$).

The data set can be downloaded from https://archive.ics.uci.edu/dataset/732/darwin:

```
temp <- tempfile()
download.file("https://archive.ics.uci.edu/static/public/732/darwin.zip", temp)
darwin_tmp <- read.csv(unzip(temp,  "data.csv"), stringsAsFactors = TRUE)
```

Before proceeding with the analysis, the data is screened for multivariate outliers using the DDC algorithm in package **cellWise**.

```
darwin_orig <- list(
  x = as.matrix(darwin_tmp[, !(colnames(darwin_tmp) %in% c("ID", "class"))]),
  y = as.numeric(darwin_tmp$class) - 1)

tmp <- cellWise::DDC(darwin_orig$x,
                     list(returnBigXimp = TRUE,
                          tolProb = 0.999,
                          silent = TRUE))
```

```
 The final data set we will analyze has 174 rows and 446 columns.
```

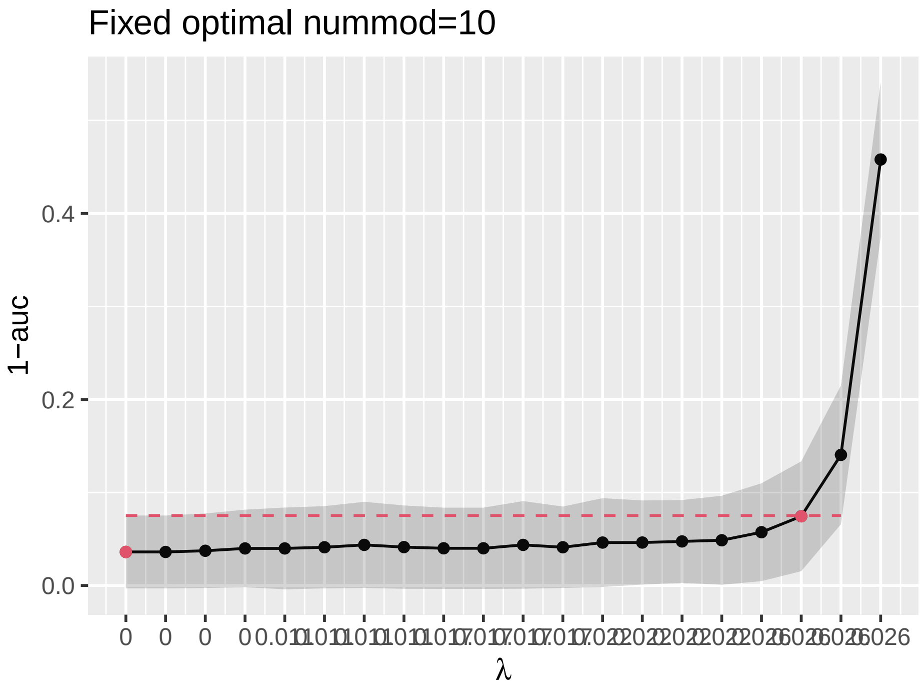
```
darwin <- list(x = tmp$Ximp,
               y = darwin_orig$y)
```

We estimate the spar model with binomial family and logit link and use 1−area under the ROC curve as the cross-validation measure:

```
spar_darwin <- spar.cv(darwin$x, darwin$y,
                       family = binomial(logit),
                       nummods = c(5, 10, 20, 50),
                       type.measure = "1-auc")
```

The `plot` method for ‘`spar.cv`’ objects displays by default the measure employed in the cross validation (in this case MSE) for a grid of λ values, where the number of models is fixed to the value found to perform best in cross-validation exercise:

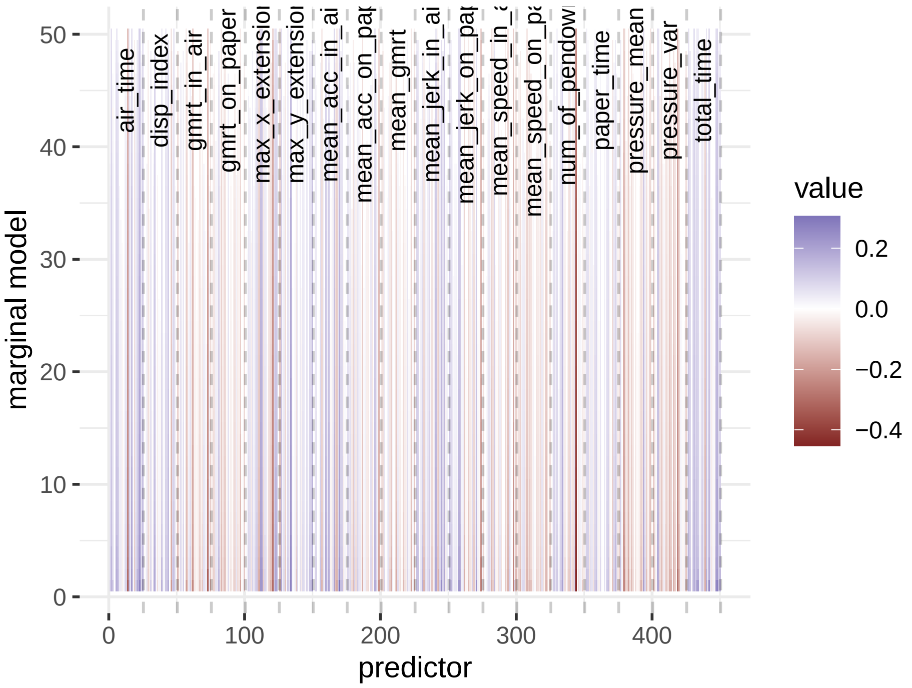
```
plot(spar_darwin)
```



The plot of the coefficients can be interpreted nicely in this example:

```
ntasks <- 25
nfeat <- 18
reorder_ind <- c(outer((seq_len(ntasks) - 1) * nfeat, seq_len(nfeat), "+"))
```

```
feat_names <- sapply(colnames(darwin$x)[seq_len(nfeat)],
                     function(name) substr(name, 1, nchar(name) - 1))

plot(spar_darwin,"coefs",coef_order = reorder_ind) +
  geom_vline(xintercept = 0.5 + seq_len(ntasks - 1) * ntasks,
             alpha = 0.2, linetype = 2) +
  annotate("text",x = (seq_len(nfeat) - 1) * ntasks + 12,
           y = 45,label=feat_names, angle = 90,
           size = 3)
```



In general we observe that the different features measures across different tasks have the same impact on the probability of AD (observable by the blocks of blue or red lines).

5. Conclusion

Package **SPAR** provides an implementation for estimating an ensemble of GLMs after performing probabilistic screening and random projection in a high-dimensional setting.

Computational details

The results in this paper were obtained using R 4.2.1.

R itself and all packages used are available from the Comprehensive R Archive Network (CRAN) at https://CRAN.R-project.org/.

Acknowledgments

Roman Parzer and Laura Vana-Gür acknowledge funding from the Austrian Science Fund (FWF) for the project "High-dimensional statistical learning: New methods to advance economic and sustainability policies" (ZK 35), jointly carried out by WU Vienna University of Economics and Business, Paris Lodron University Salzburg, TU Wien, and the Austrian Institute of Economic Research (WIFO).

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. Special Issue on PODS 2001, URL https://www.sciencedirect.com/science/article/pii/S0022000003000254.

Aghila G, Siddharth R (2020). ***RandPro**: Random Projection with Classification.* R package version 0.2.2, URL https://CRAN.R-project.org/package=RandPro.

Ailon N, Chazelle B (2009). "The fast Johnson–Lindenstrauss transform and approximate nearest neighbors." *SIAM Journal on computing*, **39**(1), 302–322.

Banerjee T, Mukherjee G, Radchenko P (2017). *fusionclust: Clustering and Feature Screening using L1 Fusion Penalty.* R package version 1.0.0, URL https://CRAN.R-project.org/package=fusionclust.

Cannings TI, Samworth RJ (2021). ***RPEnsemble**: Random Projection Ensemble Classification.* R package version 0.5, URL https://CRAN.R-project.org/package=RPEnsemble.

Cheng X, Wang H, Zhu L, Zhong W, Zhou H (2024). *MFSIS: Model-Free Sure Independent Screening Procedures.* R package version 0.2.1, URL https://CRAN.R-project.org/package=MFSIS.

Cho H, Fryzlewicz P (2012). "High dimensional variable selection via tilting." *Journal of the Royal Statistical Society Series B: Statistical Methodology*, **74**(3), 593–622.

Cho H, Fryzlewicz P (2016). *tilting: Variable Selection via Tilted Correlation Screening Algorithm.* R package version 1.1.1, URL https://CRAN.R-project.org/package=tilting.

Cilia ND, De Gregorio G, De Stefano C, Fontanella F, Marcelli A, Parziale A (2022). "Diagnosing Alzheimer's disease from on-line handwriting: A novel dataset and performance benchmarking." *Engineering Applications of Artificial Intelligence*, **111**, 104822. ISSN 0952-1976. doi:https://doi.org/10.1016/j.engappai.2022.104822. URL https://www.sciencedirect.com/science/article/pii/S0952197622000902.

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*, p. 81–90.

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.

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.

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.

Fan J, Lv J (2007). "Sure Independence Screening for Ultra-High Dimensional Feature Space." *J Roy Stat Soc*, **B 70**.

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`. URL https://doi.org/10.1214/10-AOS798.

Fang YH, Wang JH, Hsiung CA (2017a). "TSGSIS: a high-dimensional grouped variable selection approach for detection of whole-genome SNP–SNP interactions." *Bioinformatics*, **33**(22), 3595–3602. ISSN 1367-4803. `doi:10.1093/bioinformatics/btx409`. https://academic.oup.com/bioinformatics/article-pdf/33/22/3595/50307208/bioinformatics_33_22_3595.pdf, URL https://doi.org/10.1093/bioinformatics/btx409.

Fang YH, Wang JH, Hsiung CA (2017b). *TSGSIS: Two Stage-Grouped Sure Independence Screening.* R package version 0.1, URL https://CRAN.R-project.org/package=TSGSIS.

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:https://doi.org/10.1016/0095-8956(88)90043-3`. URL https://www.sciencedirect.com/science/article/pii/0095895688900433.

Gataric M, Wang T, Samworth RJ (2019). ***SPCAvRP**: Sparse Principal Component Analysis via Random Projections (SPCAvRP).* R package version 0.4, URL https://CRAN.R-project.org/package=SPCAvRP.

Hu W, Huang M, Pan W, Wang X, Wen C, Tian Y, Zhang H, Zhu J (2024). *cdcsis: Conditional Distance Correlation Based Feature Screening and Conditional Independence Inference.* R package version 2.0.4, URL https://CRAN.R-project.org/package=cdcsis.

Johnson W, Lindenstrauss J (1984). "Extensions of Lipschitz maps into a Hilbert space." *Contemporary Mathematics*, **26**, 189–206. `doi: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`. URL https://doi.org/10.1214/23-EJS2150.

Li D, Chakraborty D, Dutta S, Roy V (2024). *bravo: Bayesian Screening and Variable Selection.* R package version 3.2.1, URL https://CRAN.R-project.org/package=bravo.

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, p. 287–296. Association for Computing Machinery, New York, NY, USA. ISBN 1595933395. `doi:https://doi.org/10.1145/1150402.1150436`. URL https://doi.org/10.1145/1150402.1150436.

Li P, Hastie TJ, Church KW (2007). "Nonlinear estimators and tail bounds for dimension reduction in L1 using Cauchy random projections." *Journal of Machine Learning Research*, **8**(Oct), 2497–2532.

Li R, Huang L, Dziak J (2022). *VariableScreening: High-Dimensional Screening for Semiparametric Longitudinal Regression.* R package version 0.2.1, URL https://CRAN.R-project.org/package=VariableScreening.

Li R, Zhong W, Zhu L (2012). "Feature screening via distance correlation learning." *Journal of the American Statistical Association*, **107**(499), 1129–1139.

Ma X, Zhang J (2016). "Robust model-free feature screening via quantile correlation." *Journal of Multivariate Analysis*, **143**, 472–480.

Ma X, Zhang J, Zhou J (2015). *QCSIS: Sure Independence Screening via Quantile Correlation and Composite Quantile Correlation.* R package version 0.1, URL https://CRAN.R-project.org/package=QCSIS.

Mai Q, Zou H (2013). "The Kolmogorov filter for variable screening in high-dimensional binary classification." *Biometrika*, **100**(1), 229–234. URL https://doi.org/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`. URL https://doi.org/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`.

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:https://doi.org/10.1080/01621459.2019.1677240`. https://doi.org/10.1080/01621459.2019.1677240, URL https://doi.org/10.1080/01621459.2019.1677240.

Parzer R, Filzmoser P, Vana-Gür L (2024). "Sparse Data-Driven Random Projection in Regression for High-Dimensional Data." `doi:https://doi.org/10.48550/arXiv.2312.00130`. 2312.00130.

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:https://doi.org/10.18637/jss.v083.i02`.

Shin M, Bhattacharya A, Johnson VE (2017). "Scalable Bayesian Variable Selection Using Nonlocal Prior Densities in Ultrahigh-Dimensional Settings." 1507.07106, URL https://arxiv.org/abs/1507.07106.

Shin M, Tian R (2020). *BayesS5: Bayesian Variable Selection Using Simplified Shotgun Stochastic Search with Screening (S5)*. R package version 1.41, URL https://CRAN.R-project.org/package=BayesS5.

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. URL https://www.sciencedirect.com/science/article/pii/S2352711020303423.

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, Cham. ISBN 978-3-319-41573-4. doi:https://doi.org/10.1007/978-3-319-41573-4_3. URL https://doi.org/10.1007/978-3-319-41573-4_3.

Tian Y, Feng Y (2021). *RaSEn: Random Subspace Ensemble Classification and Variable Screening*. R package version 3.0.0, URL https://CRAN.R-project.org/package=RaSEn.

van Rossum G, *et al.* (2011). *Python Programming Language*. URL http://www.python.org.

Wang R, Dutta S, Roy V (2021). "Bayesian iterative screening in ultra-high dimensional settings." 2107.10175, URL https://arxiv.org/abs/2107.10175.

Wang X, Leng C (2016). "High-dimensional Ordinary Least-squares Projection for Screening Variables." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **78**, 589–611. doi:https://doi.org/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.

Wanjun Liu Yuan Ke JL, 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. https://doi.org/10.1080/01621459.2020.1783274, URL https://doi.org/10.1080/01621459.2020.1783274.

Wu M, Li Y, Li R (2022). *LqG: Robust Group Variable Screening Based on Maximum Lq-Likelihood Estimation*. R package version 0.1.0, URL https://CRAN.R-project.org/package=LqG.

Xu C, Chen J (2014). "The sparse MLE for ultrahigh-dimensional feature screening." *Journal of the American Statistical Association*, **109**(507), 1257–1269.

Zang Q, Xu C, Burkett K (2020). *SMLE:An R Package for Joint Feature Screening in Ultrahigh-dimensional GLMs.*

Zhu J, Pan W, Zheng W, Wang X (2021). "**Ball**: An R Package for Detecting Distribution Difference and Association in Metric Spaces." *Journal of Statistical Software*, **97**(6), 1–31. doi:10.18637/jss.v097.i06.

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.

Affiliation:

Roman Parzer
Computational Statistics (CSTAT) Institute of Statistics and Mathematical Methods in Economics
Karlsplatz 4
Vienna Austria
E-mail: Roman.Parzers@tuwien.ac.at

Peter Filzmoser

Laura Vana Gür

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
MMMMMM YYYY, Volume VV, Issue II *Submitted:* yyyy-mm-dd
doi:10.18637/jss.v000.i00 *Accepted:* yyyy-mm-dd