Introduction to Penalization methods (Lasso, Ridge, SCAD, SIR, Dantzig Selector) and some screening methods SIS, SAFE

The methods we study here will primarily be applied to the linear regression model. Penalization methods are not restricted to be applied only to linear models. They can be sufficiently modified for other types of models too like sparse PCA, CCA, nonlinear regressions, etc. For simplicity, all the methods will be illustrated for linear models only. Hence, our model is,

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon},$$

where $\mathbf{y} = (y_1, \ldots, y_n)$, $\mathbf{X}$ is an $n \times p$ dimensional design matrix and $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)$ is a vector of length p.

1 Regression models

Least square: This is the most popular estimation algorithm for linear regression. The objective function is given by

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \frac{1}{n}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2,$$

where $\|\cdot\|_2$ stands for the ℓ_2 norm i.e. $\|\mathbf{a}\|_2^2 = \sum_{i=1}^p a_i^2$. The solution of the above minimization problem is $\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$. However, this will only work if $\mathbf{X}^T\mathbf{X}$ is invertible.

Hence, least square is an issue when $n < p$ or $\mathbf{X}$ has linearly dependent columns. For the first situation, regularization methods provide better solutions. The common theme of regularization methods is that they try to shrink the parameter values to zero with the help of some penalty term. Then the regression will look like

$$\frac{1}{n}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \sum_{j=1}^{p} \text{pen}_\lambda(|\beta_j|). \tag{1}$$

Here, the function $\text{pen}_\lambda : [0, \infty) \to [0, \infty)$ can be taken to be the LASSO penalty (absolute value) or one of its variants like the ALASSO, AEN, or SCAD etc.

Ridge: The objective function is given by,

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \frac{1}{n}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_2^2,$$

where the part $\lambda\|\boldsymbol{\beta}\|_2^2$ is called the Ridge penalty. For Ridge penalty, we have a closed form solution for $\boldsymbol{\beta}$ i.e. $\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X} + n\lambda\mathbf{I}_p)^{-1}\mathbf{X}^T\mathbf{y}$.

Observe that, $\frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2$ is minimized when $\mathbf{y} \approx \mathbf{X}\boldsymbol{\beta}$. On the other hand $\lambda\|\boldsymbol{\beta}\|_2^2$ is minimized when $\boldsymbol{\beta} = 0$. Thus, when we attempt to minimize them together, $\boldsymbol{\beta}$ will simultaneously try to satisfy $\mathbf{y} \approx \mathbf{X}\boldsymbol{\beta}$ and $\boldsymbol{\beta} \approx 0$. This will kind of make components in $\boldsymbol{\beta}$ very close or exactly equal to zero if the corresponding predictor does not have any significant 'linear' effect on response. Given the nature of the closed-form solution, this penalty is also useful when $\mathbf{X}^T\mathbf{X}$ is singular, i.e. contains zero eigen-values (thus not invertible).

Least Absolute Shrinkage and Selection Operator (LASSO): The objective function is given by,

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}}(\frac{1}{n}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1),$$

where the part $\lambda\|\boldsymbol{\beta}\|_1$ is called the LASSO penalty [8]. Here $\|\boldsymbol{\beta}\|_1 = \sum_{j=1}^{p}|\beta_j|$ where $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)$.

Ridge estimates have lower bias in estimates, but greater prediction error. However, LASSO estimates have higher bias, but lower prediction error. Thus elastic net penalty was proposed to bring a balance between the two.

[12, 5, 10] introduced and further developed Debiased LASSO to reduce the noise due to LASSO estimation.

Elastic net: The objective function is given by,

$$\hat{\boldsymbol{\beta}} \equiv \arg\min_{\boldsymbol{\beta}}(\frac{1}{n}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda(1-\alpha)\|\boldsymbol{\beta}\|_2^2 + \lambda\alpha\|\boldsymbol{\beta}\|_1.$$

The elastic net method includes the LASSO and ridge regression: in other words, each of them is a special case where $\alpha = 1$ or $\alpha = 0$, respectively.

When the variables in $\mathbf{X}$ are highly correlated, the elastic net is preferred over LASSO.

Fitting a LASSO, Ridge and Elastic-net There are several algorithms to fit a LASSO and Ridge regression.

```
set.seed(1234)
N = 500
p = 30
X = scale(matrix(rnorm(N*p), ncol=p))
b = c(.5, -.5, .25, -.25, .125, -.125, rep(0, 24))
```

```
y = scale(X %*% b + rnorm(N, sd=.5))

ytrain <- y[1:400]
Xtrain <- X[1:400, ]

ytest <- y[401:500]
Xtest <- X[401:500, ]

lambda = 0.1
#betaridge <- solve(crossprod(X)+lambda*diag(p)) %*% crossprod(X, y)
fit <- glmnet::glmnet(Xtrain, ytrain, alpha = 0, intercept = F, lambda = lambda)
betaridge <- fit$beta

fit <- glmnet::glmnet(Xtrain, ytrain, alpha = 1, intercept = F, lambda = lambda)
betalasso <- fit$beta

sum((b-betaridge)^2)
sum((b-betalasso)^2)

#Biases
abs(b[1:6]-betaridge[1:6])
abs(b[1:6]-betalasso[1:6])

#Predicted mean
yhatridge <- Xtest %*% betaridge
yhatlasso <- glmnet::predict.glmnet(fit, Xtest) #This simply computes Xtest %*% betalasso

sum((ytest-yhatridge)^2)
sum((ytest-yhatlasso)^2)
```

However, in practice, it is difficult to ascertain a 'correct' value of λ. Hence, cross-validation based methods are considered to select λ based on some pre-specified criteria.

```
#Cross-validation based
CVr <- glmnet::cv.glmnet(Xtrain, ytrain, alpha = 0, intercept = F)
fitr <- glmnet::glmnet(Xtrain, ytrain, alpha = 0, intercept = F, lambda = CVr$lambda.min)
betaridge <- fitr$beta
```

```
CVl <- glmnet::cv.glmnet(Xtrain, ytrain, alpha = 1, intercept = F)
fitl <- glmnet::glmnet(Xtrain, ytrain, alpha = 1, intercept = F, lambda = CVl$lambda.min)
betalasso <- fitl$beta

sum((b-betaridge)^2)
sum((b-betalasso)^2)

#Predicted mean
yhatridge <- Xtest %*% betaridge
yhatlasso <- glmnet::predict.glmnet(fit, Xtest) #This simply computes Xtest %*% betalasso

sum((ytest-yhatridge)^2)
sum((ytest-yhatlasso)^2)
```

Group LASSO: The estimator is defined as [11]

$$\hat{\boldsymbol{\beta}}_\lambda = \arg\min_{\boldsymbol{\beta}} (\frac{1}{n}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda \sum_{g=1}^{G} \|\boldsymbol{\beta}_{\mathcal{I}_g}\|_2),$$

where $\mathcal{I}_g$ is the index set belonging to the g-th group of variables, $g = 1, \ldots, G$.

Why $\|\boldsymbol{\beta}_{\mathcal{I}_g}\|_2$, but not $\|\boldsymbol{\beta}_{\mathcal{I}_g}\|_2^2$ like ridge? The purpose of group LASSO is to identify important groups. We saw that for selecting important predictors, LASSO is better than the ridge. Technically, $\|\boldsymbol{\beta}_{\mathcal{I}_g}\|_2$ and $\|\boldsymbol{\beta}_{\mathcal{I}_g}\|_2^2$ induce a similar type of penalization scheme (which is due to expected correlation among the predictors within a given group). However, we consider the penalty as $\|\boldsymbol{\beta}_{\mathcal{I}_g}\|_2$ without the square. It is due to the fact that when group-sizes are all 1 i.e. the number of elements in $\mathcal{I}_g$ is 1 for all g and β_g is the only element in $\boldsymbol{\beta}_{\mathcal{I}_g}$, we have $\sum_{g=1}^{G} \|\boldsymbol{\beta}_{\mathcal{I}_g}\|_2 = \sum_{g=1}^{G} |\beta_g|$ i.e. the LASSO penalty. Hence, the above penalty induces a LASSO-type penalty across the groups. This would help our purpose of selecting important groups. Hence, the above penalty induces a within-group ridge penalty and for across-group, a LASSO-type penalty.

Sparse group LASSO The sparse group lasso estimator of $\boldsymbol{\beta}$ is given by [7]

$$\hat{\boldsymbol{\beta}}_{\lambda_1,\lambda_2} = \arg\min_{\boldsymbol{\beta}} (\frac{1}{n}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda_1\|\boldsymbol{\beta}\|_1 + \lambda_2 \sum_{g=1}^{G} \|\boldsymbol{\beta}_{\mathcal{I}_g}\|_2),$$

which adds a LASSO penalty term to the original group lasso model.

One can combine group LASSO and sparse group LASSO and re-write the objective function as

$$\hat{\boldsymbol{\beta}}_{\lambda,\gamma} = \arg\min_{\boldsymbol{\beta}}(\frac{1}{n}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \gamma\lambda\|\boldsymbol{\beta}\|_1 + \lambda\sum_{g=1}^{G}\|\boldsymbol{\beta}_{\mathcal{I}_g}\|_2)$$

with $\gamma > 0$. Here setting $\gamma = 0$, it would be plain group LASSO.

or,

One can combine group LASSO and sparse group LASSO and re-write the objective function as

$$\hat{\boldsymbol{\beta}}_{\lambda,\alpha} = \arg\min_{\boldsymbol{\beta}}(\frac{1}{n}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \alpha\lambda\|\boldsymbol{\beta}\|_1 + (1-\alpha)\lambda\sum_{g=1}^{G}\|\boldsymbol{\beta}_{\mathcal{I}_g}\|_2)$$

with $1 \geq \alpha \geq 0$. Here setting $\alpha = 0$, it would be plain group-LASSO.

Example of sparse group LASSO

[4] has a nice biological application of sparse group-LASSO method. In this paper, y_i is flowering time of *i-th* flower, x_i is the SNP-data of *i-th* flower. The groups $\mathcal{G}$s' are different genes. In case, you do not know. Each gene corresponds a set of SNPs. Thus, in studies to select 'important' SNPs for a given phenotype y_i, it is reasonable to consider a sparse group LASSO method taking the SNPs corresponding to the same gene as groups. More specifically, the index set $\mathcal{I}_g$ would be the SNP indices corresponding to *g-th* gene. With more mutations, one gene corresponds to several SNPs.

Fused LASSO: Check the following example.

```
dim1 = dim2 = 16
p = dim1*dim2
n = 300
X = matrix(rnorm(n*p),nrow=n)
beta0 = matrix(0,dim1,dim2)
beta0[(row(beta0)-dim1/2)^2 + (col(beta0)-dim2/2)^2 <=
        (min(dim1,dim2)/3)^2] = 1
fields::image.plot(beta0)
y = X %*% as.numeric(beta0) + rnorm(n)

# Takes about 30 seconds for the full solution path
```

```
out = genlasso::fusedlasso2d(y,X,dim1=dim1,dim2=dim2)

# Grab the solution at 8 values of lambda over the path
a = coef(out,nlam=8)

# Plot these against the true coefficients
oldpar <- par(no.readonly = TRUE)
on.exit(par(oldpar))
par(mar=c(1,1,2,1),mfrow=c(3,3))

cols = terrain.colors(30)
zlim = range(c(range(beta0),range(a$beta)))
image(beta0,col=cols,zlim=zlim,axes=FALSE)

for (i in 1:8) {
  image(matrix(a$beta[,i],nrow=dim1),col=cols,zlim=zlim,
        axes=FALSE)
  mtext(bquote(lambda==.(sprintf("%.3f",a$lambda[i]))))
}

#Cross-validation based
CVr <- glmnet::cv.glmnet(X, y, alpha = 0, intercept = F)
fitr <- glmnet::glmnet(X, y, alpha = 0, intercept = F, lambda = CVr$lambda.min)
betaridge <- fitr$beta

CVl <- glmnet::cv.glmnet(X, y, alpha = 1, intercept = F)
fitl <- glmnet::glmnet(X, y, alpha = 1, intercept = F, lambda = CVl$lambda.min)
betalasso <- fitl$beta

fields::image.plot(matrix(betaridge, dim1))
fields::image.plot(matrix(betalasso, dim1))
sum((matrix(betaridge, dim1)-beta0)^2)
sum((matrix(betalasso, dim1)-beta0)^2)

fields::image.plot(matrix(a$beta[,8],nrow=dim1))
sum((a$beta[,8]-beta0)^2)
```

In the above simulation experiment, fused LASSO produced better estimates as it incorporated the 'extra' information that the neighboring locations in beta0 had small differences

in values. Specifically, we minimized the following loss applying the function `fusedlasso2d`,

$$\frac{1}{2}\|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\|_2^2+\gamma\lambda\|\boldsymbol{\beta}\|_1+\lambda\sum_{(i,k)\neq(j,\ell)|i-j|+|k-\ell|<2}|\beta_{i,k}-\beta_{j,\ell}|,$$

We can see that the coefficient $\boldsymbol{\beta}$ has some continuity. Specifically $|\beta_{i,k}-\beta_{i,k+1}|$, $|\beta_{i,k}-\beta_{i+1,k}|$, $|\beta_{i,k}-\beta_{i,k-1}|$, $|\beta_{i,k}-\beta_{i-1,k}|$ are in general very small for most pairs (i,j). Thus, taking inspiration from the LASSO penalty, we should here penalize these differences instead. [9] introduced this method. In the true, beta0, we also have several locations to be zero. Hence, we also add a LASSO penalty like sparse group LASSO. In the above loss-function, $\lambda\sum_{(i,k)\neq(j,\ell)|i-j|+|k-\ell|<2}|\beta_{i,k}-\beta_{j,\ell}|$ is the fused penalty part and $\gamma\lambda\|\boldsymbol{\beta}\|_1$ is the LASSO penalty part. With $\gamma=0$, it is an only fused-penalty regression.

Modifications for 1D or any dimensional coefficient array is straightforward. The main idea here is to penalize the differences in coeffients at any two neighboring locations.

Smoothly Clipped Absolute Deviations (SCAD): For SCAD, we set penalty function as,

$$\text{pen}_\lambda(x) \quad = \begin{cases} \lambda|x| & \text{if } |x|\leq\lambda, \\ \frac{2\lambda\gamma|x|-x^2-\lambda^2}{2(\gamma-1)} & \text{if } \lambda<|x|\leq\gamma\lambda, \\ \frac{\lambda^2(\gamma+1)}{2} & \text{if } |x|>\gamma\lambda, \end{cases}$$

for $\gamma>2$.

Note that SCAD is the same as the lasso until $|x|=\lambda$, then smoothly transitions to a quadratic function until $|x|=\gamma\lambda$, after which it remains constant for all $|x|>\gamma\lambda$. It was introduced in [2].

Minimax Concave Penalty (MCP):
For MCP, we set penalty function as,

$$\text{pen}_\lambda(x) \quad = \begin{cases} \lambda|x|-\frac{x^2}{2\gamma} & \text{if } |x|\leq\lambda\gamma, \\ \frac{\lambda^2\gamma}{2} & \text{if } |x|>\gamma\lambda, \end{cases}$$

for $\gamma>1$.

Comparison between SCAD and MCP can be found at `https://myweb.uiowa.edu/pbreheny/7600/s16/notes/2-29.pdf`.

Note that SCAD and MCP are alternative to LASSO. Due to computational issues, they are less popular than LASSO. But still there are a lot of merits. For coefficient of any magnitude $\frac{\partial \text{pen}_{\lambda,\text{LASSO}}(|\beta_j|)}{\lambda} = \text{sign}(\beta_j)\lambda$. However for SCAD and MCP, $\frac{\partial \text{pen}_{\lambda,\text{SCAD or MCP}}(|\beta_j|)}{\lambda} = 0$ for $|\beta_j| > \gamma\lambda$. Thus, the effect of Penalty is no longer there in the rate of change of overall loss with respect to the coefficient. As a consequnce, this reduces the bias in the estimated coefficients. LASSO estimates in general have greater bias. To tackle this issue under LASSO framework, adaptive LASSO was proposed in [13]. Here the penalty is again proposed to be coefficient's magnitude specific. The penalty for adaptive LASSO is $\lambda \sum_{j=1}^{p} w_j|\beta_j|$. The weights are assigned following different schemes. Weights are often pre-specified or changed adaptively throughout the optimization scheme. `https://myweb.uiowa.edu/pbreheny/7600/s16/notes/2-29.pdf` has some discussions on this. We often run the optimization with the usual LASSO penalty $\lambda \sum_{j=1}^{p} |\beta_j|$ first, say for 1000 iterations. Thereafter, we can set $w_j = \frac{1}{\beta_j^{(1000)}}$ and start running the adaptive LASSO. For w_j, we can actually set any nonincreasing function of $\beta_j^{(1000)}$. Another option to fit the adaptive LASOO is the following.

---

1) Getting weights $\mathbf{w}$: Set $\mathbf{w} = 1/|\hat{\boldsymbol{\beta}}_A|^{\zeta}$ where $\hat{\boldsymbol{\beta}}_A = \arg\min_{\boldsymbol{\beta}} \frac{1}{2n}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda_R\|\boldsymbol{\beta}\|_2^2$ with λ_R being selected via cross-validation. Choices of $\zeta = 0.5, 1, 2$ [13]. If $n > p$, the ridge estimation can be replaced by regular OLS.

2) A-LASSO Fitting: $\frac{1}{2n}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\mathbf{w} \circ \boldsymbol{\beta}\|_1$ using `cv.glmnet` with penalty.factor $= \mathbf{w}$.

```
require(glmnet)

## Ridge Regression to create the Adaptive Weights Vector
set.seed(999)
cv.ridge <- cv.glmnet(x, y, alpha=0, parallel=TRUE, standardize=TRUE)
w <- 1/abs(coef(cv.ridge, s=cv.ridge$lambda.min))^1 ## Using zeta = 1
w[w == Inf] <- 999999999 ## Replacing values estimated as Infinite for 999999999

## Adaptive Lasso
set.seed(999)
cv.lasso <- cv.glmnet(x, y, alpha=1, parallel=TRUE, standardize=TRUE, penalty.factor=w)
plot(cv.lasso)
plot(cv.lasso$glmnet.fit, xvar="lambda", label=TRUE)
```

```
abline(v = log(cv.lasso$lambda.min))
abline(v = log(cv.lasso$lambda.1se))
coef(cv.lasso, s=cv.lasso$lambda.1se)
coef <- coef(cv.lasso, s='lambda.1se')
```

Dantzig selector: Assume that the setup is the same as in the previous LASSO example. When $p > n$, it is usually possible to get a $\boldsymbol{\beta}$ such that $\mathbf{y} = \mathbf{X}\boldsymbol{\beta}$. But to avoid over-fitting, we don't want exact equality. This means $\mathbf{X}\beta \approx \mathbf{y}$. This leads to a linear program known as the Dantzig Selector with tuning parameter λ,

$$\min_{\boldsymbol{\beta}:\|\mathbf{X}^T(\mathbf{y}-\mathbf{X}\beta)\|_\infty \leq \lambda} \|\boldsymbol{\beta}\|_1,$$

which is due to [1]. The norm $\|\mathbf{a}\|_\infty = \max(\mathbf{a})$. The solution of the above matches with the LASSO solution

2 Screening

Screening rules helps to pre-compute the variables that are bound to be zero in the optimal solution. Then discarding these variables makes it easier to solve the problem. A lot of research are motivated on designing screening rules for different high dimensional regression problem. R package `VariableScreening` is a great resource. This package has a nice description on different screening methods. Dr Jianqin Fan and Dr Runze Li have several papers on different screening methods.

If conditional distribution of y given the predictors $x = (x_1, ..., x_p)$ i.e. $P(y \mid x)$ functionally depends on x_k, then y and x_k are usually marginally dependent as well. Hence, a relatively moderate-size set of variables can be constructed by selecting only the predictors that are marginally dependent with y, and this procedure is commonly known as independence screening. Furthermore, such a screening procedure is referred to as a *sure independence screening method* when it can be shown to identify all the important predictors in a supervised learning setting, with probability tending to 1 as sample size increases [3].

Sure independence screening (SIS): We will see Vanilla-SIS. It is very simple to understand. We run un-normalized maximization without any penalty. Then pick $\mathcal{A} = \{j : |\beta_j| > \delta\}$, for some threshold, δ which often depends on the number of samples n as well as the number of predictors p. [6] has a good tutorial type paper on this method. After this

preselection, we run our penalized regression with only the predictors in $\mathcal{A}$.

SAfe Feature Elimination (SAFE): It is based on the following result.

Theorem 1. *SAFE rule for the lasso problem: For any $i \in \{1, \ldots, p\}$, if the following is satisfied:*

$$|\mathbf{x}_i^T \mathbf{y}| < \lambda - sd(\mathbf{x}_i) sd(\mathbf{y}) \frac{\lambda_{\max} - \lambda}{\lambda_{\max}}$$

where $\lambda_{\max} = \|\mathbf{X}^T \mathbf{y}\|_\infty / n$, then in the optimal solution $\boldsymbol{\beta}$, we have $\beta_i = 0$.

Hence, for all the predictors, we can check the above condition, and only include those predictors for the follow-up penalized regression that do not have zero in the optimal solution.

3 Software programs for implementation

R packages:

`glmnet` is an excellent package to implement LASSO regression. `https://glmnet.stanford.edu/articles/glmnet.html` has excellent tutorial on using this package. `ncvreg` is an excellent package for SCAD. `genlasso` is another great package. In addition, there are few other packages for group LASSO.

In python, most of the available regressors are in `sklearn` package itself. If I have time, I will go over the `sklearn` package later.

The issue with these packages is that they are mostly written for continuous responses like the above examples. For other types of variables like binary or discrete, the user need to write their own objective function and use different optimizers in most cases. For non-continuous response, a standard implementation may not be very stable. It might require a lot of tuning to get a reasonably good estimate. Hence, standard packages often do not support that beyond binary data.

References

[1] Emmanuel Candes and Terence Tao. The dantzig selector: Statistical estimation when p is much larger than n. *The Annals of Statistics*, 35(6):2313–2351, 2007.

[2] Jianqing Fan and Runze Li. Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American statistical Association*, 96(456):1348–1360, 2001.

[3] Jianqing Fan and Jinchi Lv. Sure independence screening for ultrahigh dimensional feature space. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 70(5):849–911, 2008.

[4] Yingjie Guo, Chenxi Wu, Maozu Guo, Quan Zou, Xiaoyan Liu, and Alon Keinan. Combining sparse group lasso and linear mixed model improves power to detect genetic variants underlying quantitative traits. *Frontiers in Genetics*, 10:271, 2019.

[5] Adel Javanmard and Andrea Montanari. Confidence intervals and hypothesis testing for high-dimensional regression. *The Journal of Machine Learning Research*, 15(1):2869–2909, 2014.

[6] Diego Franco Saldana and Yang Feng. Sis: An r package for sure independence screening in ultrahigh-dimensional statistical models. *Journal of Statistical Software*, 83:1–25, 2018.

[7] Noah Simon, Jerome Friedman, Trevor Hastie, and Robert Tibshirani. A sparse-group lasso. *Journal of computational and graphical statistics*, 22(2):231–245, 2013.

[8] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society B*, 58:267–288, 1996.

[9] Robert Tibshirani, Michael Saunders, Saharon Rosset, Ji Zhu, and Keith Knight. Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(1):91–108, 2005.

[10] Sara Van de Geer, Peter Bühlmann, Ya'acov Ritov, and Ruben Dezeure. On asymptotically optimal confidence regions and tests for high-dimensional models. 2014.

[11] Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006.

[12] Cun-Hui Zhang and Stephanie S Zhang. Confidence intervals for low dimensional parameters in high dimensional linear models. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 76(1):217–242, 2014.

[13] Hui Zou. The adaptive lasso and its oracle properties. *Journal of the American statistical association*, 101(476):1418–1429, 2006.