# cossonet: An Integrated R Package for Fast Sparse Nonparametric Regression for High-dimensional data

*by Jieun Shin and Changyi Park*

**Abstract** The package **cossonet** offers the practical function of COSSO, a nonparametric regression model based on ANOVA decomposition. **cossonet** has several improvements for general usability in high-dimensional data. The package **cossonet** eliminates matrix inversion in the COSSO algorithm by employing the coordinate descent algorithm to fast computation. It supports various response variables belonging to the exponential family, and it is extended with the elastic-net penalty to select correlated components more effectively. We show simulation and real datasets analysis to varify its usability.

## 1 Introduction

With the rapid development of data collection technology, the amount and complexity of data in many fields have increased exponentially. In high-dimensional data which is the number of predictors $d$ exceeds the number of observations $n$, one of the goals of statistical modeling is to identify complex relationships between response and explanatory variables. One simple approach to achieve this goal is linear regression, but it does not easily provide well-defined coefficient estimates (Montgomery et al., 2021). In high-dimensional data, model selection is essential because in many cases there are only a few variables that affect predictive performance. Statistical variable selection methods such as forward selection, backward elimination, and stepwise procedures are basic approaches, while regularization methods such as LASSO (Tibshirani, 1996), ridge (Hoerl and Kennard, 1970), SCAD (Fan and Li, 2001), and elastic-net (Zou and Hastie, 2005) are also widely used.

Unfortunately, the expectation that explanatory variables linearly explain the response variable is often unrealistic. Nonparametric models are good alternative because they allow nonlinear relationships and are more flexible than parametric models. Several widely used nonparametric models, such as CART (Breiman et al., 1984), MARS (Friedman, 1991), BRUTO (Hastie and Stuetzle, 1989), and TURBO (Friedman and Silverman, 1989), employ greedy algorithms to find local optima, similar to the forward selection and backward elimination of linear regression.

Smoothing spline ANOVA (SS-ANOVA) is a widely used method for multivariate function estimation (Wahba, 1990, Gu:2013). Recently, Lin and Zhang (2006) proposed the COmponent Selection and Smoothing Operator (COSSO), a nonparametric regression model that performs model selection and estimation simultaneously, and is applied in various fields (Kavuri and Kokjohn, 2017, Dempsey:2017). COSSO introduces a new LASSO-type penalty that consists of the sum of the norms of the components. The COSSO penalty is computationally useful for high-dimensional data because the ANOVA decomposition approximates high dimensions as a sum of low dimensions. This penalty estimates a sparse component similar to LASSO, which is a global optimum. Lin and Zhang (2006) showed that LASSO is a special case of the COSSO penalty. After Lin and Zhang (2006) introduced COSSO for Gaussian regression, Zhang and Lin (2006) and Leng and Zhang (2006) extended it to the exponential family and the Cox proportional hazards (CoxPH) model, respectively.

In this paper, we introduce **cossonet** which is an integrated R package for fast sparse nonparametric regression for high-dimensional data. **cossonet** is a function integrating models of Zhang and Lin (2006), Lin and Zhang (2006), and Leng and Zhang (2006). **cossonet** provides three key contributions. First, the standard LASSO-type penalty is extended to an elastic-net type to select correlated components. Second, **cossonet** aims to achieve fast computation for $n$ and $d$. The existing studies derived algorithms to solve the inverse matrix for $n$ and $d$. To reduce the computational burden for large $n$, they used an alternative algorithm based on subset of $m < n$. **cossonet** is implemented as a coordinate descent algorithm under the subset-based algorithm for fast computations for both large $n$ and $d$. Finally, **cossonet** can deal with a various response types including continuous, binary classes, non-negative counts, and survival. A comparison for the R package **cossonet** and **cosso** can be seen in Table 1.

This paper is organized as follows. Section `\\@ref(sec2)` provides an introduction to the SS-ANOVA framework, which serves as the foundation for the COSSO model, and defines COSSO with elastic-net penalty for the exponential family and CoxPH models. In Section @(sec3), coordinate descent algorithms are derived for the regression models, along with a description of smoothing spline selection via cross-validation. Section 2.4 presents the usage of the **cossonet** package and analyzes

**Table 1:** Comparison of the key capabilities of the cossonet and cosso packages.

| Response Type | cossonet | cosso | LASSO | Ridge | Elastic net | main e |
|---|---|---|---|---|---|---|
| Gaussian | \$\checkmark\$ | \$\checkmark | \$\checkmark | \$\checkmark | \$\checkmark\$ | \$\chec |
| Binomial | \$\checkmark\$ | \$\checkmark\$ | \$\checkmark\$ | \$\checkmark\$ | \$\checkmark\$ | \$\chec |
| Poisson | \$\checkmark\$ | \$\times\$ | \$\times\$ | \$\times\$ | \$\times\$ | \$\chec |
| Cox | \$\checkmark\$ | \$\checkmark\$ | \$\checkmark\$ | \$\checkmark\$ | \$\checkmark\$ | \$\chec |

simulation results obtained with **cossonet**. Real data analysis and discussion of the results are included in Section 2.5. Finally, Section **??** provides the conclusion of the paper.

## 2 Model

### Smoothing Spline ANOVA

The functional ANOVA decomposition expresses a multivariate function $f$ as a sum of components, capturing different levels of interactions among the explanatory variables. For explanatory variables $x = (x^{(1)}, \dots, x^{(d)}) \in \mathcal{X} = [0,1]^d$, the decomposition is represented as

$$f(x) = f_0 + \sum_{j=1}^{d} f_j(x^{(j)}) + \sum_{j<k} f_{jk}(x^{(j)}, x^{(k)}) + \cdots + f_{1,\dots,d}(x^{(1)}, \dots, x^{(d)}), \tag{1}$$

where $f_0$ is a constant term, $f_j$'s are the main effects, and $f_{jk}$'s are two-way interactions, and so on. The identifiability of the terms in equation (1) is ensured by certain side conditions. To simplify the model, SS-ANOVA typically considers only lower-order interactions, such as main effects and two-way interactions.

Each main effect $f_j(x^{(j)})$ belongs to a reproducing kernel Hilbert space (RKHS) $\mathcal{H}^{(j)}$ which is decomposed as

$$\mathcal{H}^{(j)} = \mathcal{H}_0^{(j)} \oplus \mathcal{H}_1^{(j)}, \tag{2}$$

where $\mathcal{H}_0^{(j)}$ is the mean space and $\mathcal{H}_1^{(j)}$ is the contrast space. If $x^{(j)}$ is continuous, $\mathcal{H}^{(j)}$ is often considered as the second-order Sobolev Hilbert space

$$W_2[0,1] = \{g : g \text{ and } g' \text{ are absolutely continuous, and } g'' \in \mathcal{L}_2[0,1]\}.$$

The inner product with respect to $W_2[0,1]$ is defined as

$$< f, g >= \left[\int_0^1 g(t)\,dt\right]^2 + \left[\int_0^1 g'(t)\,dt\right]^2 + \int_0^1 [g''(t)]^2\,dt,$$

and the corresponding reproducing kernel is

$$K(s,t) = 1 + k_1(s)k_1(t) + k_2(s)k_2(t) - k_4(|s-t|),$$

where $k_1(s) = s - 1/2$, $k_2(s) = (k_1^2(s) - 1/12)/2$, and $k_4(t) = (k_1^4(t) - k_1^2(t)/2 + 7/240)/24$. For derivation of the reproducing kernel for the continuous variable, see Chapter 2.3 of Gu and Gu (2013). Suppose that categorical variables $x^{(j)}$ is taken $L$ distinct values from the set $\{1, \dots, L\}$ and $f_j(x^{(j)})$ is represented as an $L$-vector. The space $\mathcal{H}_0^{(j)}$ is defined as $\{f : f(1) = \cdots = f(L)\}$ and $\mathcal{H}_1^{(j)}$ is defined as $\{f : f(1) + \cdots + f(L) = 0\}$. The reproducing kernel for $\mathcal{H}_1^{(j)}$ in equation (2) is

$$K_1(s,t) = L \cdot I(s=t) - 1,$$

where $I(\cdot)$ is the indicator function. For the derivation of the reproducing kernel for the category variable, see Chapter 2.2 of Gu and Gu (2013).

The full space corresponding to the decomposition (1) is expressed as a tensor product

$$\otimes_{j=1}^{d} \mathcal{H}^{(j)} = \{1\} \oplus \left[\oplus_{j=1}^{d} \mathcal{H}_1^{(j)}\right] \oplus \left[\oplus_{j<k} \mathcal{H}_1^{(j)} \otimes \mathcal{H}_1^{(k)}\right] \oplus \cdots. \tag{3}$$

If $x^{(j)}$ is continuous, the reproducing kernel of the tensor product space $\mathcal{H}^{(j)} \otimes \mathcal{H}^{(k)}$ is the product of $K(s^{(j)}, t^{(j)})$ and $K(s^{(k)}, t^{(k)})$. If $x^{(j)}$ is categorical, the reproducing kernel is then the product of $K_1(s^{(j)}, t^{(j)})$ and $K_1(s^{(k)}, t^{(k)})$. Thus, the full space $\mathcal{F}$ corresponding to equation (3) is

$$\mathcal{F} = \{1\} \oplus_{v=1}^{p} \mathcal{F}^{(v)}, \tag{4}$$

where $\mathcal{F}^{(1)}, \ldots, \mathcal{F}^{(p)}$ are $p$ orthogonal subspaces of $\mathcal{F}$. For the main effects model, $p = d$ and each $\mathcal{F}^{(v)}$ corresponds to a subspace of the main effects. This case is equivalent to an additive model. For the two-way interactions model, $p = d(d+1)/2$ and $\mathcal{F}^{(v)}$ include subspaces of the main effects and the two-way interactions.

## COSSO penalized likelihood

The COSSO models first derived under the Gaussian regression framework (Lin and Zhang, 2006) and the exponential family (Zhang and Lin, 2006) and CoxPH regression model (Leng and Zhang, 2006). These models can be implemented using the **cosso** package, which supports continuous, binary class, and survival responses. Our package extends the capabilities of the **cosso** package to support non-negative count responses. Consider a regression problem where $Y_i = f(x_i) + \epsilon_i, i = 1, \ldots, n$ with $x_i \in [0, 1]$ and $\epsilon \sim N(0, \sigma^2)$, where $Y$ represents the response value corresponding to $x$. The COSSO penalized likelihood for estimating $f$ is defined by

$$-\ell(f(x), Y) + \tau^2 J(f), \tag{5}$$

where $\ell$ is the log-likelihood function and $\tau > 0$ is the smoothing parameter, and $J(f)$ is the roughness penalty.

The log-likelihood $\ell$ is determined by the type of response variable. We first consider the exponential family. Assume that the response variable $Y$ follows a distribution from the exponential family with the density

$$\exp\left\{\frac{yf(x) - B(f(x))}{A(\sigma)} + D(y, \sigma)\right\},$$

where $A > 0, B$ and $D$ are known functions, and $\sigma$ is a nuisance parameter. From the properties of the exponential family, the conditional mean of $Y$ given $X = x$ is $\mathrm{E}(Y|x) = \dot{B}(f(x)) = \mu(x)$, and the conditional variance is $\mathrm{Var}(Y|x) = \ddot{B}(f(x))A(\sigma) = \nu(x)A(\sigma)$, where $\dot{B}$ and $\ddot{B}$ denote the first and second derivatives of $B$, respectively. The log-likelihood for the exponential family is given by

$$\ell(f(x), y) = \sum_{i=1}^{n} \{y_i f(x_i) - B(f(x_i))\}. \tag{6}$$

The cases below describe the types of responses considered in our model.

### Case 1. Gaussian regression

Assume that the response variables follow a Gaussian distribution, $Y_i|x_i \sim N(f(x_i), \sigma^2)$. One has $A(\sigma) = \sigma^2, B(f) = f^2/2$, and $D(y, \sigma) = -y^2/(2\sigma^2)$.

### Case 2. Logistic regression

Consider a binary response variable, where $Y_i \in \{0, 1\}$ and the probability of $Y_i = 1$ given $x_i$ is $P(Y_i = 1|x_i) = \pi(x_i)$, such that $\log \pi(x_i)/(1 - \pi(x_i)) = f(x_i)$. One has $A(\sigma) = 1, B(f) = \log(1 + \exp(f))$, and $D(y, \sigma) = 1$.

### Case 3. Poisson regression

For the nonnegative count response $Y \in \{0, 1, 2, \ldots\}$, assumed to follow a Poisson distribution with $P(Y_i = y_i|x_i) = \lambda^{y_i} e^{-\lambda_i}/(y_i!)$, where $\log \lambda_i = f(x_i)$. One has $A(\sigma) = 1, B(f) = \exp(f)$, and $D(y, \sigma) = -\log(y!)$.

### Case 4. CoxPH regression

We next consider of survival data analysis. Let the censored times be $Y = \min\{T, C\}$, where $T$ is the event time and $C$ is the censoring time. Assume that $T$ and $C$ are conditionally independent given $X = x$ and censoring mechanism is independent. Define $\delta = I(T \leq C)$ as the event indicator. The observed data consists of the triplet $\{Y_i, \delta_i, x_i\}_{i=1}^n$.

For simplicity, we assume that the observed failure times are distinct. When failure times are tied, the technique proposed by Breslow (1974) can be applied. Let $t_1 < \cdots < t_N$ be the unique and ordered failure times. For the observed data, the risk set right before $t_j$ is defined as

$$R_j = \{i \mid y_i \geq t_j\},$$

where $R_j$ includes all observations $i$ still at risk at $t_j$. The hazard function given $x$ is

$$h(t|x) = h_0(t)\exp(f(x)),$$

where $h_0$ is an unspecified baseline hazard function. The partial log-likelihood is then

$$\ell(f(x_i)) = \sum_{j=1}^{N} \delta_j \left[ f(x_j) - \log\left\{ \sum_{i \in R_j} \exp(f(x_i)) \right\} \right]. \tag{7}$$

## 3 Algorithm

### Equivalent formulation

Lin and Zhang (2006) proved that the minimizer of equation (5) lies on a finite-dimensional space of the RKHS $\mathcal{F}^{(v)}$. To simplify the computation, equation (5) is replaced by the following equivalent formulation. The minimizer of equation (5) is the the standard COSSO solution obtained by solving

$$\begin{aligned} &\min_{f,\theta} -\ell(f(x), y) + \lambda_0 \sum_{v=1}^{p} \theta_v^{-1} \|P^{(v)} f\|^2, \\ &\text{subject to } \sum_{v=1}^{p} \theta_v \leq M, \quad \theta_v \geq 0, \quad v = 1, \ldots, p. \end{aligned} \tag{8}$$

Here, $J(f) = \sum_{v=1}^{d} \theta_v^{-1} \|P^{(v)} f\|^2$, where $P^{(v)} f$ is the orthogonal projection of $f$ onto $\mathcal{F}^{(v)}$ and $\|\cdot\|$ denotes the RKHS norm associated with $\mathcal{F}^{(v)}$. $\lambda_0 > 0$ is a constant, and the smoothing parameter $M$ controls sparsity by constraining the sum of $\theta_v$'s, which is also regulate the solution's roughness. Equation (8) is similar to the formulation of a smoothing spline with multiple smoothing parameters and an alternative penalty on the $\theta$'s.

The elastic-net penalized COSSO extends $\theta$ in equation (8) as

$$\begin{aligned} &\min_{f,\theta} -\ell(f(x), y) + \lambda_0 \sum_{v=1}^{p} \theta_v^{-1} \|P^{(v)} f\|^2, \\ &\text{subject to } \gamma \sum_{v=1}^{p} \theta_v + (1-\gamma) \sum_{v=1}^{p} \theta_v^2 \leq M, \quad \theta_v \geq 0, \quad v = 1, \ldots, p. \end{aligned} \tag{9}$$

Equation (9) is equivalently expressed as

$$\begin{aligned} &\min_{f,\theta} -\ell(f(x), y) + \lambda_0 \sum_{v=1}^{p} \theta_v^{-1} \|P^{(v)} f\|^2 + n\lambda \left( \gamma \sum_{v=1}^{p} \theta_v + (1-\gamma) \sum_{v=1}^{p} \theta_v^2 \right), \\ &\text{subject to } \theta_v \geq 0, \quad v = 1, \ldots, p, \end{aligned} \tag{10}$$

where $\lambda > 0$ is a smoothing parameter and the mixing parameter $0 \leq \gamma \leq 1$ controls the balance between the LASSO and ridge penalties. When $\gamma = 1$, equations (9) and (10) reduce to the standard COSSO.

For fixed $\lambda_0$ and $\lambda$, the representation theorem for smoothing spline guarantees that the minimizer of equations (9) and (10) is $f(x) = \sum_{i=1}^{n} c_i K(x_i, x) + b$, where $c_i$ and $b \in \mathbb{R}$ are the smoothing spline parameters and $K(x_i, x) = \sum_{v=1}^{p} \theta_v K^{(v)}(x_i, x)$ with $K^{(v)}(x_i, x)$ being the reproducing kernels in $\mathcal{F}^{(v)}$. For simply notation, denote $K^{(v)}$ as the $n \times n$ matrix $\{K^{(v)}(x_i, x_k)\}_{i,k=1}^{n}$ and $K$ as the matrix $\{K(x_i, x)\}_{i=1}^{n}$.

Let $\boldsymbol{c} = (c_1, \ldots, c_n)^T$ and $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_p)^T$. Following the standard COSSO framework, the roughness penalty in equations (9) and (10) is given by $\sum_{v=1}^{d} \theta_v^{-1} \|P^{(v)} f\|^2 = \sum_{v=1}^{d} \theta_v \boldsymbol{c}^T K^{(v)} \boldsymbol{c} = \boldsymbol{c}^T K \boldsymbol{c}$. The matrix formulation of equation (10) becomes

$$\min_{b,\boldsymbol{c},\boldsymbol{\theta} \geq \mathbf{0}} -\frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i) + \lambda_0 \boldsymbol{c}^T K \boldsymbol{c} + \lambda(\gamma \mathbf{1}_d^T \boldsymbol{\theta} + (1-\gamma) \boldsymbol{\theta}^T \boldsymbol{\theta}). \tag{11}$$

This matrix formulation enables practical computation of equation (10). Section @ref(ssec3_2) outlines the coordinate descent algorithm for the exponential family, and Section @ref(ssec3_3) addresses the algorithm for the CoxPH regression model.

## Elastic-net based COSSO for exponential family

The basic approach to handling exponential families in the COSSO framework is to optimize $f$ by applying Newton iterative method (Gu and Gu, 2013). Our coordinate descent algorithm is derived from weighted least squares based on Newton iteration. Given the current solution $f^0(x_i)$, the conditional mean and variance of $Y$ given $x_i$ are $u_i^0 = \dot{B}(f^0(x_i))$ and $V_i^0 = \ddot{B}(f^0(x_i))$, respectively. With $\nu_i = -y_i + \mu_i^0$ and $w_i = V_i^0$, the second-order Taylor expansion of $-y_i f(x_i) + B(f(x_i))$ at $f^0(x_i)$ is expressed as

$$\begin{aligned} & -y_i f^0(x_i) + B(f^0(x_i)) + \nu_i \{f(x_i) - f^0(x_i)\} + \frac{1}{2} w_i \{f(x_i) - f^0(x_i)\}^2 \\ &= \frac{1}{2} w_i \left\{ f(x_i) - f^0(x_i) + \frac{\nu_i}{w_i} \right\}^2 + O_i, \end{aligned}$$

where $O_i$ is independent of $f(x_i)$. Let $z_i = f^0(x_i) + (y_i - \mu_i^0)/w_i$ be the working response. For matrix formulation, let $W = \text{diag}(w_1, \ldots, w_n)$ and $\mathbf{z} = (z_1, \ldots, z_n)^T$. The Newton iteration for solving (11) is equivalent to the minimizer of the following objective function

$$\min_{b,\boldsymbol{c},\boldsymbol{\theta} \geq \mathbf{0}} (\mathbf{z} - U\boldsymbol{c} - b\mathbf{1}_n)^T W (\mathbf{z} - U\boldsymbol{c} - b\mathbf{1}_n) + n\lambda_0 \boldsymbol{c}^T U \boldsymbol{c} + \lambda(\gamma \mathbf{1}_d^T \theta + (1-\gamma)\theta^T \theta). \tag{12}$$

Lin and Zhang (2006) and Zhang and Lin (2006) proposed a two-step algorithm to solve equation (12). In the first step, with $\boldsymbol{\theta}$ fixed, the solution for $(b, \boldsymbol{c})$ is obtained as the solution of a smoothing spline. In the second step, with $(b, \boldsymbol{c})$ fixed, the solution for $\boldsymbol{\theta}$ is obtained as the solution of a non-negative garrote. In the algorithms of Lin and Zhang (2006) and Zhang and Lin (2006), the inverse matrix is needed to solve these two steps. However, the **cosso** package use `glmnet` and `solve.QP` for pratical purposes. When the `ginv` function is used to calculate the Moore-Penrose generalized inverse, the computational complexity is $\mathcal{O}(n^3 + p^3)$ per iteration. Lin and Zhang (2006) proposed an alternative algorithm to reduce computational complexity for large sample sizes $n$ achieves $\mathcal{O}(nm^2 + p^3)$ per iteration. Since the subset is independent of $p$, this approach still offers no improvement for $p$. In contrast, our coordinate descent algorithm achieves efficiency for both $n$ and $p$ with a complexity of $\mathcal{O}(nm^2 + np)$ per iteration.

## Coordinate Descent Algorithm

Lin and Zhang (2006) showed that simulation performance of the subset-basis algorithm was comparable to that of the full-basis algorithm. However, the subset-based algorithm significantly improves computational efficiency compared to the full-basis algorithm. We solve equation (12) using a two-step subset-based approach as proposed by Lin and Zhang (2006). Suppose that we have $m \leq n$ subsets, then $\boldsymbol{c} = (c_1, \ldots, c_m)^T$. Let $Q$ be the matrix $\{K(x_{k^*}, x_{l^*})\}_{k,l=1}^{m}$ and $Q^{(v)}$ be the matrix $\{K^{(v)}(x_{k^*}, x_{l^*})\}_{k,l=1}^{m}$. Let $U$ be the matrix $\{K(x_k, x_{l^*})\}$ and $U^{(v)}$ be the matrix $\{K^{(v)}(x_k, x_{l^*})\}$ for $k = 1, \ldots, n$ and $l = 1, \ldots, m$. Let $U_{ij}$ and $Q_{ij}$ denote the $(i,j)$th entries of $U$ and $Q$, respectively. The solution to the subspace is approximated by $f(x) = \sum_{i=1}^{m} c_i \sum_{v=1}^{p} \theta_v U^{(v)} + b$. Equation (12) becomes

$$\min_{b,\boldsymbol{c},\boldsymbol{\theta} \geq \mathbf{0}} \sum_{i=1}^{n} w_i \left( z_i - \sum_{k=1}^{m} c_k U_{ik} - b \right)^2 + n\lambda_0 \sum_{k=1}^{m} \sum_{l=1}^{m} c_k c_l Q_{kl} + \lambda \left( \gamma \sum_{v=1}^{p} \theta_v + (1-\gamma) \sum_{v=1}^{p} \theta_v^2 \right). \tag{13}$$

Following Tibshirani (1996), equation (13) can be transformed into a penalized weighted least squares (WLS) problem. Solving for the WLS solution is efficient for existing algorithm by avoiding matrix inversion. Our algorithm finds the WLS solution at each step. When $\boldsymbol{\theta}$ is fixed, equation (13) simplifies to

$$\min_{b,\boldsymbol{c}} \sum_{i=1}^{n} w_i \left( z_i - \sum_{k=1}^{m} c_k U_{ik} - b \right)^2 + n\lambda_0 \sum_{k=1}^{m} \sum_{l=1}^{m} c_k c_l Q_{kl}. \tag{14}$$

This is equivalent to a weighted smoothing spline. The solutions for $b$ and $\boldsymbol{c}$ is given by

$$\begin{aligned} \hat{c}_j &= \frac{2\sum_{i=1}^{n} w_i U_{ij}(z_i - \sum_{l\neq j} U_{il}c_l - b) - n\lambda_0 \sum_{l\neq j}^{m} Q_{jl}c_l}{2\sum_{i=1}^{n} w_i U_{ij}^2 + n\lambda_0 Q_{jj}}, \quad j = 1,\ldots,m, \\ \hat{b} &= \frac{\sum_{i=1}^{n} w_i (z_i - \sum_{l=1}^{n} U_{il}c_l)}{\sum_{i=1}^{n} w_i}. \end{aligned} \tag{15}$$

When $(b, \boldsymbol{c})$ is fixed, let $u_i = \sqrt{w}_i(z_i - b)$, $G$ be the $n \times p$ matrix where the $i$th row and $v$th column element is $\sqrt{w}_i \sum_{k=1}^{m} U_{ik}^{(v)} c_k$, and $h_v$ be the vector whose $v$th element is $n\lambda_0 \sum_{k=1}^{m} \sum_{l=1}^{m} c_k c_l Q_{kl}^{(v)}$ for $v = 1,\ldots,d$. With this, equation (13) becomes

$$\min_{\boldsymbol{\theta}\geq \boldsymbol{0}} \sum_{i=1}^{n} \left( u_i - \sum_{v=1}^{p} \theta_v G_{iv} \right)^2 + \sum_{v=1}^{p} h_v \theta_v + \lambda \left( \gamma \sum_{v=1}^{p} \theta_v + (1-\gamma) \sum_{v=1}^{p} \theta_v^2 \right). \tag{16}$$

This is equivalent to a non-negative garrote with elastic-net regularization. The solution for $\boldsymbol{\theta}$ is

$$\hat{\theta}_j = \frac{\mathcal{S}\left(\sum_{i=1}^{n} G_{ij}(u_i - \sum_{k\neq j} G_{ij}\theta_j) - h_j/2, n\lambda\gamma/2\right)}{\sum_{i=1}^{n} G_{ij}^2 + n\lambda(1-\gamma)}, \quad \text{for } j = 1,\ldots,p, \tag{17}$$

where $\mathcal{S}(\alpha,\beta)$ is the soft-thresholding operator defined as

$$\begin{aligned} \mathcal{S}(\alpha,\beta) &= \text{sign}(\alpha)\,[|\alpha| - \beta]_+ \\ &= \begin{cases} \alpha - \beta, & \text{if } \alpha > 0 \text{ and } \beta < |\alpha|, \\ 0, & \text{otherwise.} \end{cases} \end{aligned}$$

Given a new data point $x^* = (x_1^*, \ldots, x_p^*)$, the function is predicted by

$$\hat{f}(x^*) = \hat{b} + \sum_{i=1}^{m} \hat{c}_i \sum_{v=1}^{p} \hat{\theta}_v U^{(v)}(x^*, x_i^*).$$

This predicted value outcomes for new data points based on the fitted model. To obtain accurate predictions, it is important to choose an appropriate smoothing parameter, which is discussed in the following sections.

## Smoothing parameter selection

The smoothing parameters $\lambda_0$ and $\lambda_\theta$ control the trade-off between the likelihood function and the penalty term. The problem of smoothing parameter selection is directly related to selecting the optimal model. $\lambda_0$ and $\lambda_\theta$ control the strength of the regularization. A higher $\lambda_\theta$ value results in more coefficient shrinkage and sparsity. In order for theta to be chosen correctly, both $\lambda_0$ and $\lambda_\theta$ must be set to a grid with an appropriate range. We choose $\lambda_0$ and $\lambda_\theta$ by cross-validation (CV) at each step.

Gu and Gu (2013) discussed performance-oriented iteration and CV as criterias for smoothing parameter selection. For CV, Zhang and Lin (2006) adopted Kullbeck-Leibler (KL) divergence which is one of the measurement for performance-oriented iteration. KL divergence has the advantage of directly evaluating the model with observed data without additional matrix calculation. Lin and Zhang (2006) adopted generalized cross-validation (GCV) in Gaussian regression framework. We adopted GCV defined as

$$\text{GCV} = \frac{n\sum_{i=1}^{n} w_i (y_i - \hat{f}_i)^2}{\{n - tr(H)\}^2}.$$

It is straightforward to derive and easy to apply within the WLS framework. To compute this, we need the degrees of freedom $\text{tr}(H)$, which is obtained as the sum of the diagonal elements of the hat matrix $H$, where $h_{ii} = U_i(U^T U + \lambda_0 Q)^{-1} U_i^T$. The complete algorithm that combining one-step update and parameter tuning is summarized below.

1. Initialize $f_i = f(x_i) = \bar{y}, \mu_i = \dot{B}(f_i), w_i = \ddot{B}(f_i)$, and $z_i = f_i - (y_i - \mu_i)/w_i$ for $i = 1,\ldots,n$. Set

$\theta_v = 1$ for $v = 1, \dots, p$, and define grids for $\lambda_0$ and $\lambda$.

2. With fixed $\hat{\theta}_1, \dots, \hat{\theta}_p$, compute $z_1, \dots, z_n$, $U$ and $Q$. Solve for $(\hat{b}, \hat{\boldsymbol{c}})$ using (15), and calculate the GCV across all $\lambda_0$ grid points. Select the optimal $\lambda_0$ and corresponding $(\hat{b}, \hat{\boldsymbol{c}})$ that minimize the average GCV.
3. With fixed $(\hat{b}, \hat{\boldsymbol{c}})$, compute $h_1, \dots, h_p$ and $G$. Solve for $\hat{\theta}_1, \dots, \hat{\theta}_p$ using (17), and calculate the GCV across all $\lambda$ grid points. Select the optimal $\lambda$ and corresponding $\hat{\theta}_1, \dots, \hat{\theta}_p$ that minimize the average GCV.
4. Repeat Steps 2 and 3 until convergence.

The 1-standard error rule can be applied to obtain a simpler model than the one derived using the above procedure. The **cossonet** package offers options to implement this rule.

## Elastic-net based COSSO for CoxPH regression model

Leng and Zhang (2006) proposed an objective function for the CoxPH regression model within the COSSO framework. We extend this model by applying an elastic-net penalty, as defined in the following objective function

$$\min_{\boldsymbol{c}, \boldsymbol{\theta} \geq \boldsymbol{0}} -\frac{1}{n} \boldsymbol{\delta}^T U \boldsymbol{c} + \frac{1}{n} \sum_{j=1}^{N} \log \left( \sum_{i \in R_j} e^{U_i \boldsymbol{c}} \right) + \lambda_0 \boldsymbol{c}^T Q \boldsymbol{c} + \lambda (\gamma \mathbf{1}_p^T \boldsymbol{\theta} + (1 - \gamma) \boldsymbol{\theta}^T \boldsymbol{\theta}), \tag{18}$$

where $\boldsymbol{\delta} = (\delta_1, \dots, \delta_n)$ is the vector of censoring indicators and $U_i$ is the $i$th row of $U$, and the remaining matrix notation follows the definitions provided in Section @ref{ssec3_1}.

Leng and Zhang (2006) derived a subset-based two-step algorithm where $\boldsymbol{c}$ and $\boldsymbol{\theta}$ are updated iteratively while keeping the other fixed. To solve $\boldsymbol{c}$ and $\boldsymbol{\theta}$, the gradient and Hessian of both are calculated. The $\boldsymbol{c}$ vector is updated using the Newton-Raphson iteration, while $\boldsymbol{\theta}$ is solved via quadratic programming with linear constraints. Since it is based on matrix inversion, the subset-based approach has a computational complexity of $\mathcal{O}(nm^2 + p^3)$. In practice, the **cosso** package employs `glmnet` and `solve.QP` to estimate $\boldsymbol{c}$ and $\boldsymbol{\theta}$, respectively. we extend the subset-based coordinate descent approach using WLS Simon et al. (2011) to the CoxPH regression model. The computation complexity becomes $\mathcal{O}(nm + np)$ per iteration.

## Coordinate Descent Algorithm

We rewrite equation (18) as the observed objective function

$$\min_{\boldsymbol{c}, \boldsymbol{\theta} \geq \boldsymbol{0}} -\frac{1}{n} \sum_{j=1}^{N} \delta_j \left( U_j \boldsymbol{c} + \log \left( \sum_{i \in R_j} e^{U_i \boldsymbol{c}} \right) \right) + n\lambda_0 \sum_{k=1}^{m} \sum_{l=1}^{m} c_k c_l Q_{kl} + n\lambda \left( \gamma \sum_{v=1}^{p} \theta_v + (1 - \gamma) \sum_{v=1}^{p} \theta_v^2 \right), \tag{19}$$

where the first term sums over the non-censored rows of $U$. Following Simon et al. (2011), equation (19) can be transformed into a WLS formulation. Solving for the WLS solution is more efficient than existing algorithm by avoiding direct calculation the gradient and Hessian for $\boldsymbol{c}$ and $\boldsymbol{\theta}$. Our algorithm easily finds a solution as the WLS solution at each step. For simplicity of calculation, assume that the intercept is absorbed through data scaling. When $\boldsymbol{\theta}$ is fixed, equation (19) reduces to (14) where the weight and working response are given by

$$w_i = \sum_{j=1}^{N} \left[ \frac{\sum_{i \in R_j} U_i^T U_i e^{U_i \boldsymbol{c}}}{\sum_{i \in R_j} e^{U_i \boldsymbol{c}}} - \frac{\sum_{i \in R_j} U_i^T e^{U_i \boldsymbol{c}}}{\sum_{i \in R_j} e^{U_i \boldsymbol{c}}} \frac{\sum_{i \in R_j} U_i^T e^{U_i \boldsymbol{c}}}{\sum_{i \in R_j} e^{U_i \boldsymbol{c}}} \right],$$

$$z_i = f_i + \frac{1}{w_i} \left[ \delta_i + \sum_{j=1}^{N} \frac{\sum_{i \in R_j} U_i^T e^{U_i \boldsymbol{c}}}{\sum_{i \in R_j} e^{U_i \boldsymbol{c}}} \right],$$

respectively. These can be easily computed using the XX function in `glmnet`. The weight $w_i$ in **cosso** relates to the Hessian of $\boldsymbol{c}$, which involves $n^2$ calculation and is computationally intensive. However, our algorithm only needs $n$ calculations. The solution for $\boldsymbol{c}$ is equivalent to (15) without intercept. When $(b, \boldsymbol{c})$ is fixed, let $u_i = \sqrt{w_i} z_i$, $G$ be the $n \times p$ matrix where the $i$th row and $v$th column element is $\sqrt{w_i} \sum_{k=1}^{m} U_{ik}^{(v)} c_k$, and $h_v$ be the vector whose $v$th element is $(n/\lambda_0) \sum_{k=1}^{m} \sum_{l=1}^{m} c_k c_l Q_{kl}^{(v)}$ for $v = 1, \dots, d$. With this, equation (19) becomes (16). This is equivalent to the non-negative garrote with elastic-net regularization. The solution for $\boldsymbol{\theta}$ is equivalent to equation (17). Given a new data point $x^* = (x_1' \dots, x_p^*)$, the predicted function is obtained by

$$\hat{f}(x^*) = \sum_{i=1}^{m} \hat{c}_i \sum_{v=1}^{p} \hat{\theta}_v U^{(v)}(x^*, x_i^*).$$

In the CoxPH regression model, Leng and Zhang (2006) used approximate cross-validation based on KL divergence for parameter tuning. However, this measure is evaluated by the Hessian matrix obtained during the update, which is not required in our algorithm. Instead, we apply GCV within the WLS framework, following the full algorithm described in Section @ref(ssec3_2).

## 4 Example usage

**cossonet** is available from the Comprehensive R Archive Network. The main algorithm of the **cossonet** package is written in `C` and calls `C` code from `R` using `.Call()`. The main function `cossonet` accepts all of the response types we consider and returns an object of the S3 class `cossonet`. This section provides examples of how to use the **cossonet** package. We first load the library for **cossonet** and set a seed for reproducibility.

```
devtools::install_github("jiieunshin/cossonet")
library(cossonet)
set.seed(20250101)
```

### Data generation

The function `data_generation` generates example datasets with response specified as continuous, binary, non-negative count, or survival data. The detailed process of generating datasets is explained in Section @ref{sec5}. We generate a training set with $n = 200$ and $p = 20$, and a test set with $n = 200$ and $p = 20$.

```
tr = data_generation(n = 200, p = 20, SNR = 8, response = "regression",)
str(tr)

#> List of 3
#>  $ x: num [1:200, 1:20] 0.377 0.579 0.239 0.572 0.877 ...
#>  $ f: num [1:200] 2.9493 0.0907 5.6039 -0.8799 0.331 ...
#>  $ y: num [1:200] 2.9493 0.0907 5.6039 -0.8799 0.331 ...

te = data_generation(n = 1000, p = 20, SNR = 8, response = "regression")
str(te)

#> List of 3
#>  $ x: num [1:1000, 1:20] 0.873 0.619 0.416 0.431 0.205 ...
#>  $ f: num [1:1000] -0.0916 0.2171 -1.5693 -1.0491 4.1775 ...
#>  $ y: num [1:1000] -0.0916 0.2171 -1.5693 -1.0491 4.1775 ...
```

### 4.2 Model fitting

The function `cossonet` takes an $n \times p$ matrix and a response vector as input to fit the model. The `family` argument specifies the response type, defaulting to `gaussian` if not provided. It also includes arguments for model fitting, such as the kernel function and grids for smoothing parameters. If not specified, default values are used. See Table 2 for details on the arguments. We demonstrate `cossonet` with the simplest settings.

```
fit = cossonet(tr$x, tr$y, family = "gaussian",
              lambda0 = exp(seq(log(2^{-6}), log(2^{-3}), length.out = 20)),
              lambda_theta = exp(seq(log(2^{-8}), log(2^{-4}), length.out = 20))
              )
```

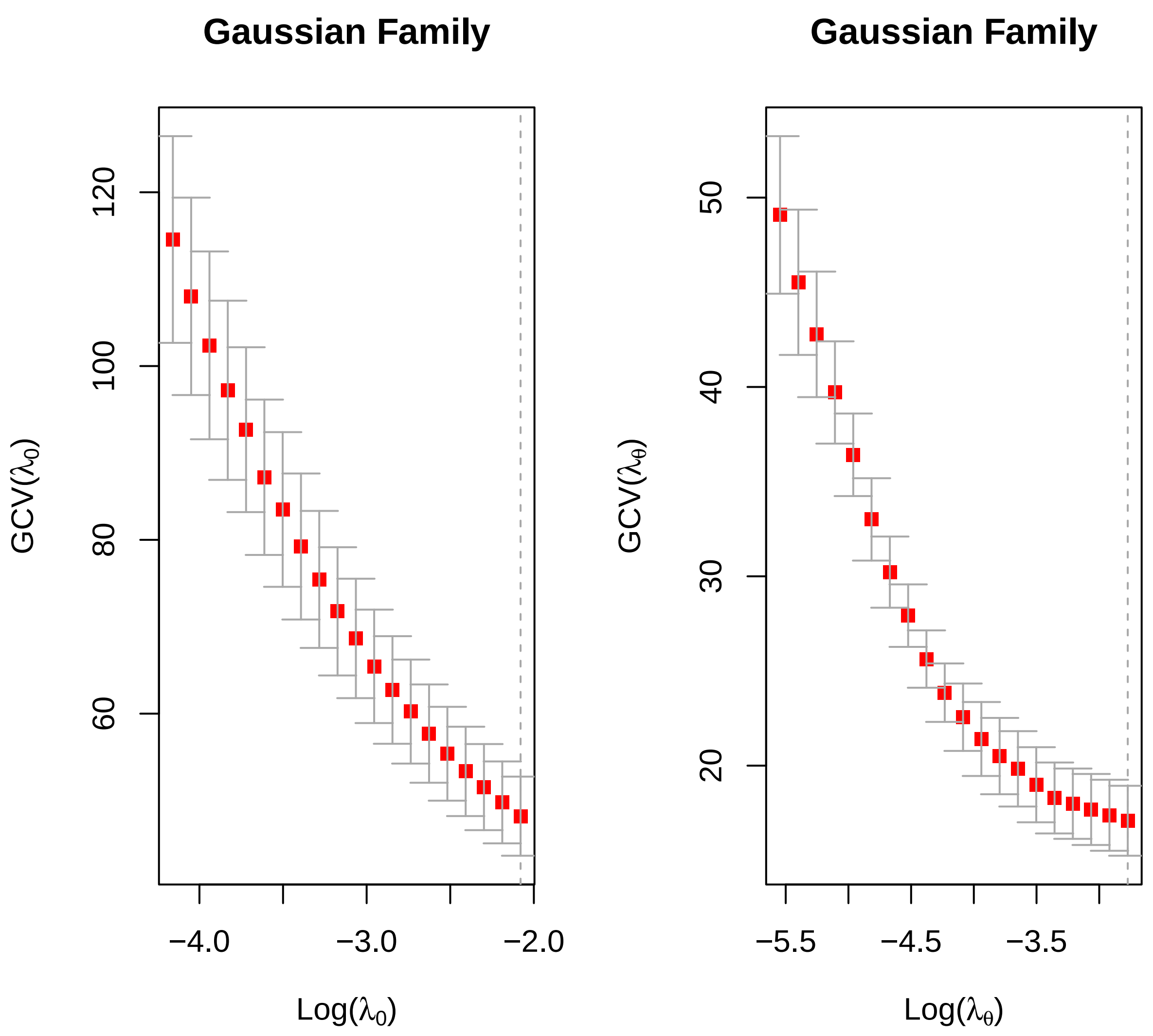


**Figure 1:** The five-fold CV plot for $\lambda_0$ and $\lambda$ from the `cossonet` run is shown. The horizontal axis represents the log of $\lambda$, and the vertical axis represents the GCV. The red points are the average GCV for the validation set, while the vertical solid lines are the standard errors. If the 1-standard error rule is applied, the smoothing parameter is selected as the value corresponding to the dotted line in the current figure.

```
#> fit COSSO  with n =  200 p = 20
#> kernel: spline and d = 20
#> -- c-step --
#> proceeding...

#> mse: 2.3582
#>
#> -- theta-step --
#> proceeding...


#> mse: 2.5687
#>
#> -- c-step --
#> proceeding...
#> mse: 2.3582
```

Running the code above displays the algorithm's progress and the error for validation set at current step in the console. Figure 1 presents the GCV curve for the validation error during 5-fold CV at each step. The red dot is the average GCV and the vertical solid line represents the standard error.

```
str(fit)
```

**Table 2:** Summary of the arguments of functions in ‘cossonet‘.

| Argument | Description |
|---|---|
| x | Input matrix of size $n$ by $p$, where each row represents an observation. It can be a matrix or da |
| y | The response variable. If \texttt{family="gaussian"} or \texttt{family="poisson"} (non-negative cou |
| family | The type of the response variable. |
| wt | The weights of the predictors. The default is \texttt{rep(1, ncol(x))}. |
| scale | If \texttt{TRUE}, continuous predictors are rescaled to the interval $[0, 1]$. The default is \texttt{T |
| cv | A measurement for cross-validation. |
| nbasis | The number of "knots" to choose from. If \texttt{basis.id} is provided, it is ignored. |
| basis.id | An index that specifies the selected "knot". |
| kernel | The kernel function. Four types are provided: \texttt{linear}, \texttt{gaussian}, \texttt{poly}, and \ |
| effect | The effect of the component. \texttt{main} (default) for the main effect, \texttt{interaction} for two |
| kparam | Parameter $\kappa$ for the kernel function. Used by Gaussian and polynomial kernels. |
| lambda0 | A vector of $\lambda_0$ sequences. The default is a grid of 20 values $\mathtt{[2^{-10}, \dots, 2^ |
| lambda | A vector of $\lambda$ sequences. The default is a grid of 20 values $\mathtt{[2^{-10}, \dots, 2^{1 |
| gamma | Elastic mesh mixing parameter, $0 \leq \gamma \leq 1$. When \texttt{gamma=1}, it uses LASSO |

```
#> List of 5
#>  $ data      :List of 7
#>   ..$ x       : num [1:200, 1:20] 0.345 0.568 0.192 0.56 0.897 ...
#>   ..$ y       : num [1:200] 2.9493 0.0907 5.6039 -0.8799 0.331 ...
#>   ..$ Uv     : num [1:200, 1:40, 1:20] 0.02699 -0.00961 0.04974 -0.00836 -0.06566 ...
#>   ..$ basis.id: int [1:40] 10 16 17 19 33 34 35 36 43 45 ...
#>   ..$ wt      : num [1:20] 1 1 1 1 1 1 1 1 1 1 ...
#>   ..$ kernel  : chr "spline"
#>   ..$ kparam  : num 1
#>  $ tune      :List of 3
#>   ..$ lambda0     : num [1:20] 0.0156 0.0174 0.0194 0.0217 0.0242 ...
#>   ..$ lambda_theta: num [1:20] 0.00391 0.00452 0.00523 0.00605 0.007 ...
#>   ..$ gamma       : num 0.95
#>  $ c_step    :List of 12
#>   ..$ measure  : num [1:5, 1:20] 102.3 72.4 115.2 112.9 175.2 ...
#>   ..$ Uv      : num [1:200, 1:40, 1:20] 0.02699 -0.00961 0.04974 -0.00836 -0.06566 ...
#>   ..$ Q        : num [1:40, 1:40] 1.7046 0.2071 -0.0312 0.1282 0.0832 ...
#>   ..$ w.new    : num [1:200] 1 1 1 1 1 1 1 1 1 1 ...
#>   ..$ sw.new   : num [1:200] 1 1 1 1 1 1 1 1 1 1 ...
#>   ..$ mu.new   : num [1:200] 1.154 0.322 2.98 0.293 -0.953 ...
#>   ..$ z.new    : num [1:200] 2.9493 0.0907 5.6039 -0.8799 0.331 ...
#>   ..$ zw.new   : num [1:200] 2.9493 0.0907 5.6039 -0.8799 0.331 ...
#>   ..$ b.new    : num 0.963
#>   ..$ c.new    : num [1:40] 0.448 1.387 -1 -0.783 0.247 ...
#>   ..$ optlambda: num 0.125
#>   ..$ conv     : logi TRUE
#>  $ theta_step:List of 4
#>   ..$ cv_error       : num [1:5, 1:20] 62.3 46.6 50.9 49.3 36.3 ...
#>   ..$ optlambda_theta: num 0.0625
#>   ..$ gamma          : num 0.95
#>   ..$ theta.new      : num [1:20] 0 0.442 1.437 1.158 0 ...
#>  $ family    : chr "gaussian"
#>  - attr(*, "class")= chr "cdcosso"
```

The fitted model includes the data used for model fitting and the estimated parameters at each step. The `data` list shows the training set input to `cossonet` and related information. The `tune` list displays the values for the tuning parameters. The `c_step` list shows the outputs related for solving $(b, \boldsymbol{c})$, and the `theta_step` list shows the outputs related for solving $\boldsymbol{\theta}$. The `family` list shows the inputted response argument. The `theta.new` in the `theta_step` list is the estimated component. If `theta.new` is greater than 0, it is considered significant component. The example below shows that the first four components are significant while the remaining components are not.

```
fit$theta_step$theta.new
```

```
#>  [1] 0.000000 0.441998 1.437467 1.157956 0.000000 0.000000 0.000000 0.000000
#>  [9] 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
#> [17] 0.000000 0.000000 0.000000 0.000000
```

### Prediction

The function `cossonet.predict` provides the predicted values for new data using the fitted model. The output of the function `cossonet.predict` includes the predicted values $\hat{f}$ from `f.new` and $\hat{\mu}$ from `mu.new` for the new data. The predicted values for the test set using our fitted model are as follows.

```
pred = cossonet.predict(fit, te$x)
str(pred)
```

```
#> List of 2
#>  $ f.new : num [1:1000] -0.343 1.06 0.149 0.945 2.702 ...
#>  $ mu.new: num [1:1000] -0.343 1.06 0.149 0.945 2.702 ...
```

In the gaussian family, since `f.new` represents the response variable, the mean squared error for evaluating test set can be calculated as follows.

```
mean((te$y - pred$f.new)^2)
```

```
#> [1] 3.120764
```

## 5 Simulation

In this section, we simulate the prediction and component selection performance of **cossonet** for all response types considered in this paper. The simulation data generation follows the methodology from Zhang and Lin (2006), Lin and Zhang (2006), and Leng and Zhang (2006). For simulation, we call the function 'data_generation to generate simulation data in **cossonet**.

### Simulation settings

Let us generate the $p$ explanatory variables $X = (X_1, \ldots, X_p)$ in the range of $[0, 1]$. The first four $X_1, \ldots, X_4$ are informative variables, which are generated from the structure $X_j = (W_j + tV)/(1 + t)$, where $W_1, \ldots, W_p$ and $V$ are i.i.d. variables generated from Uniform$(0, 1)$. The informative variables $X_j$ and $X_k, j \neq k$ are correlated as $\text{Cor}(X_j, X_k) = t^2/(1 + t^2)$. When $t = 0$, there are no correlation. The remaining $X_5, \ldots, X_p$ are i.i.d and follow Uniform$(0, 1)$. The true function $f$ is nonlinear function constructed by the informative variables. The true function is generated slightly differently for each response type. Details are described below.

### Case 1: Continuous response

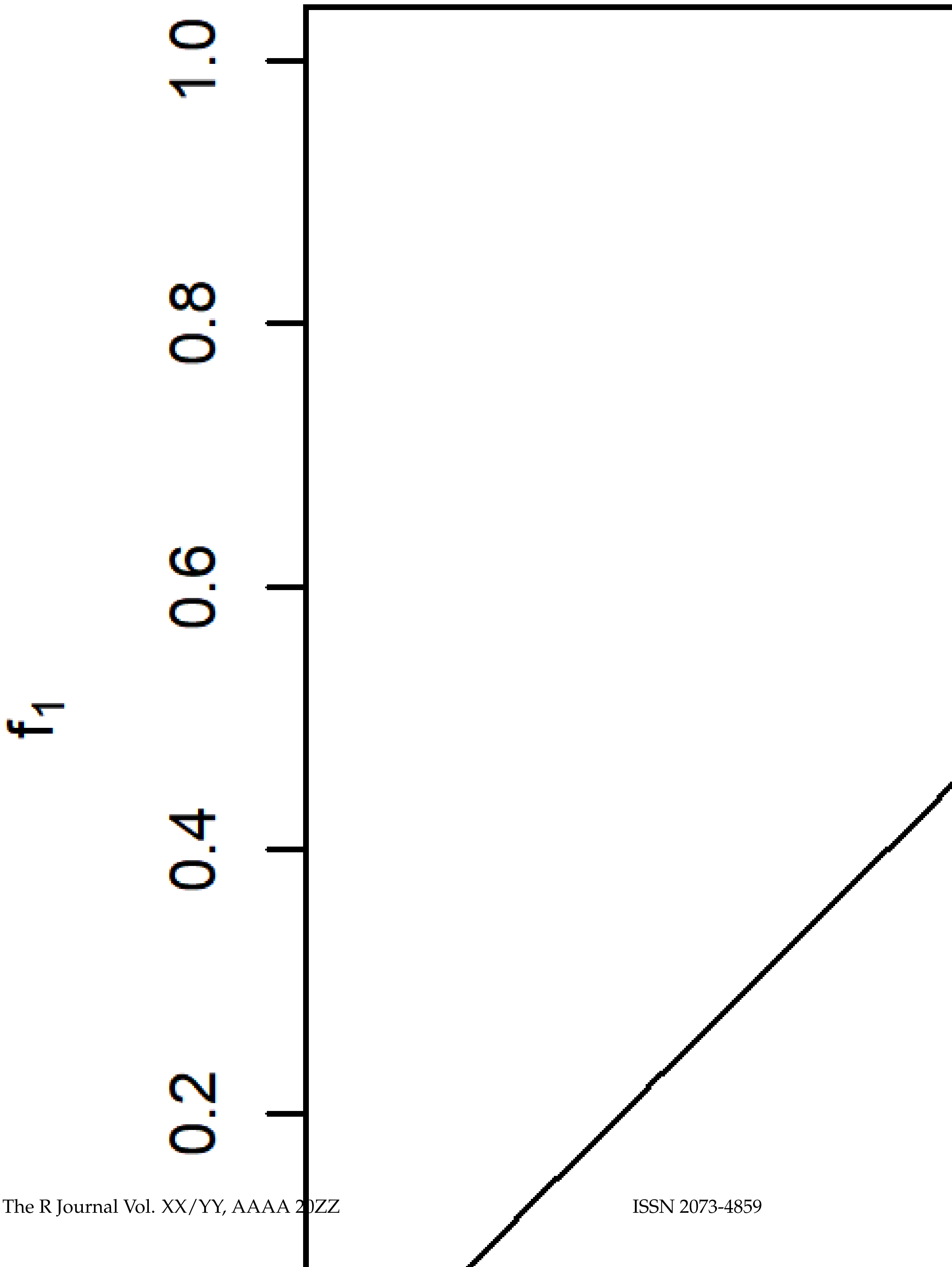
The continuous response is generated according to the procedure outlined in Zhang and Lin (2006). Consider the following nonlinear functions

$$\begin{aligned}
f_1(t) &= t \\
f_2(t) &= (2t-1)^2, \\
f_3(t) &= \frac{\sin(2\pi t)}{2-\sin(2\pi t)}, \\
f_4(t) &= 0.1\sin(2\pi t) + 0.2\cos(2\pi t) + 0.3\sin(2\pi t)^2 + 0.4\cos(2\pi t)^3 + 0.5\sin(2\pi t)^3.
\end{aligned}$$

The forms of these nonlinear functions are shown in Figure 2. The true function for the continuous response is given by

$$f(x) = f_1(x_1) + f_2(x_2) + 2f_3(x_3) + 3f_4(x_4) + \varepsilon, \tag{20}$$

where the error term $\varepsilon$ is generated from a normal distribution $N(0, \sigma)$ and $\sigma$ is chosen to have a signal-to-noise ratio of 8:1. The smoothing parameters $\lambda_0$ and $\lambda$ are considered grids of 20 values $[2^{-5}, \ldots, 2^0]$ and $[2^{-10}, \ldots, 2^{-5}]$ on an equally spaced logarithmic scale, respectively.


1.0
0.8
0.6
0.4
0.2
$f_1$

### Case 2: Binary response

The true nonlinear logit function for classification, as proposed by Lin and Zhang (2006), is given by

$$f(x) = 3x_1 + \pi \sin(\pi x_2) + 8x_3^3 + \frac{2}{e-1}e^{x_4} + \varepsilon.$$

The logit form is given by $\log \frac{\pi(x)}{1-\pi(x)} = f(x)$, which generates a binary class of 0 or 1, with the probability $\pi(x) = P(X = 1)$ in Bernoulli trials. The error $\varepsilon$ follows $N(0, \sigma)$, where $\sigma$ is chosen to have a signal-to-noise ratio of 8:1. For the smoothing parameters $\lambda_0$ and $\lambda$, we consider grids of 20 values $[2^{-6}, \ldots, 2^{-4}]$ and $[2^{-8}, \ldots, 2^{-4}]$ on an equally spaced logarithmic scale, respectively.

### Case 3: Count response

The count response is generated from a Poisson distribution with mean $\mu = \exp(f)$. The true function $f$ is generated using equation (20). To ensure the true function remains positive, appropriate values are added to the nonlinear functions: $f_5(t) = f_3(t) + 0.5$ and $f_6(t) = f_3(t) + 0.5$. The true function is then given by

$$f(x) = \frac{1}{3}(f_1(x_1) + f_2(x_2) + 2f_5(x_3) + 3f_6(x_4) + \varepsilon),$$

where $\varepsilon \sim N(0, \sigma)$, where $\sigma$ is chosen to have a signal-to-noise ratio of 8:1. The smoothing parameters $\lambda_0$ and $\lambda$ are considered as grids of 20 values $[2^{-5}, \ldots, 2^{0}]$ and $[2^{-8}, \ldots, 2^{-4}]$ on an equally spaced logarithmic scale, respectively.

### Case 4. Survival response

The survival response consists of set of time and state, where the experimental design follows the approach outlined by Leng and Zhang (2006). Based on the true function in equation (20), the survival time $T$ and the truncation time $C$ are generated from exponential distributions with means $\exp(f(x))$ and $V\exp(-f(x))$, where $V \sim \text{Uniform}(a, a+2)$. The parameter $a$ controls the truncation fraction. For example, $a = 2$ applies 20% truncation. The time response is given by $\min\{T_i, C_i\}$ and the state is defined as $I(T_i < C_i)$. The error term $\varepsilon$ is generated from $N(0, \sigma)$ with $\sigma$ chosen to have a signal-to-noise ratio of 8:1. The smoothing parameters $\lambda_0$ and $\lambda$ are considered grids of 20 values $[2^{-5}, \ldots, 2^{-2}]$ and $[2^{-10}, \ldots, 2^{-8}]$ on an equally spaced logarithmic scale, respectively.

## Bibliography

L. Breiman, J. Friedman, R. Olshen, and C. Stone. Classification and regression trees. 1984. [p1]

N. Breslow. Covariance analysis of censored survival data. *Biometrics*, 30(1):89–99, 1974. doi: https://doi.org/10.2307/2529620. [p4]

J. Fan and R. Li. Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American statistical Association*, 96(456):1348–1360, 2001. URL https://doi.org/10.1198/016214501753382273. [p1]

J. H. Friedman. Multivariate adaptive regression splines. *The annals of statistics*, 19(1):1–67, 1991. URL https://www.jstor.org/stable/2241837. [p1]

J. H. Friedman and B. W. Silverman. Flexible parsimonious smoothing and additive modeling. *Technometrics*, 31(1):3–21, 1989. URL https://www.jstor.org/stable/1270359. [p1]

C. Gu and C. Gu. *Counting processes and survival analysis*, volume 297. Smoothing spline ANOVA models, 2013. [p2, 5, 6]

T. Hastie and W. Stuetzle. Principal curves. *Journal of the American statistical association*, 84(406):502–516, 1989. URL https://www.jstor.org/stable/2289936. [p1]

A. E. Hoerl and R. W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970. URL https://www.jstor.org/stable/1271436. [p1]

C. Kavuri and S. L. Kokjohn. Computational optimization of a reactivity controlled compression ignition (rcci) combustion system considering performance at multiple modes simultaneously. *Fuel*, 207:702–718, 2017. doi: https://doi.org/10.1016/j.fuel.2017.06.071. [p1]

C. Leng and H. H. Zhang. Model selection in nonparametric hazard regression. *Nonparametric Statistics*, 18(7-8):417–429, 2006. URL https://doi.org/10.1080/10485250601027042. [p1, 3, 7, 8, 11, 13]

Y. Lin and H. H. Zhang. Component selection and smoothing in multivariate nonparametric regression. *The Annals of Statistics*, 34(5):2272–2297, 2006. URL http://www.jstor.org/stable/25463508. [p1, 3, 4, 5, 6, 11, 13]

D. C. Montgomery, E. A. Peck, and G. G. Vining. *Introduction to linear regression analysis*. John Wiley & Sons, 2021. [p1]

N. Simon, J. Friedman, T. Hastie, , and R. Tibshirani. Regularization paths for cox's proportional hazards model via coordinate descent. *Journal of statistical software*, 39(5):1–13, 2011. URL https://doi.org/10.18637/jss.v039.i05. [p7]

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 58(1):267–288, 1996. URL https://doi.org/10.1111/j.2517-6161.1996.tb02080.x. [p1, 5]

G. Wahba. *Spline models for observational data*. Society for industrial and applied mathematics, 1990. [p1]

H. H. Zhang and Y. Lin. Component selection and smoothing for nonparametric regression in exponential families. *Statistica Sinica*, 16(3):1021–1041, 2006. URL https://www.jstor.org/stable/24307585. [p1, 3, 5, 6, 11]

H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 67(2):301–320, 2005. URL https://doi.org/10.1111/j.1467-9868.2005.00527.x. [p1]

*Jieun Shin*
*University of Seoul*
*Department of Statistical Data Science*
*Seoul, Republic of Korea*
*ORCiD: 0000-1721-1511-1101*
jieunstat@uos.ac.kr

*Changyi Park*
*University of Seoul*
*Department of Statistics*
*ORCiD: 0000-0002-0912-0225*
bbil@ulm.edu