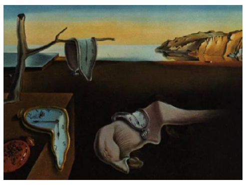

Journal of Statistical Software

June 2020, Volume 94, Issue 4. *doi: 10.18637/jss.v094.i04*

BeSS: An R Package for Best Subset Selection in Linear, Logistic and Cox Proportional Hazards Models

Canhong Wen
University of Science and Technology of China
Sun Yat-sen University

Aijun Zhang
The University of Hong Kong

Shijie Quan
Sun Yat-sen University

Xueqin Wang
University of Science and Technology of China
Sun Yat-sen University

Abstract

We introduce a new R package, **BeSS**, for solving the best subset selection problem in linear, logistic and Cox's proportional hazard (CoxPH) models. It utilizes a highly efficient active set algorithm based on primal and dual variables, and supports sequential and golden search strategies for best subset selection. We provide a C++ implementation of the algorithm using an **Rcpp** interface. We demonstrate through numerical experiments based on enormous simulation and real datasets that the new **BeSS** package has competitive performance compared to other R packages for best subset selection purposes.

Keywords: best subset selection, primal dual active set, model selection, variable selection, R, C++, **Rcpp**.

1. Introduction

One of the main tasks of statistical modeling is to exploit the association between a response variable and multiple predictors. Linear model (LM), as a simple parametric regression model, is often used to capture linear dependence between response and predictors. The other two common models: generalized linear model (GLM) and Cox's proportional hazards (CoxPH) model, can be considered as the extensions of linear model, depending on the types of responses. Parameter estimation in these models can be computationally intensive when the number of predictors is large. Meanwhile, Occam's razor is widely accepted as a heuristic rule for statistical modeling, which balances goodness of fit and model complexity. This rule leads to a relative small subset of important predictors.

The canonical approach to subset selection problem is to choose k out of p predictors for each $k \in \{0, 1, 2, \ldots, p\}$. This involves exhaustive search over all possible 2^p subsets of predictors, which is an NP-hard combinatorial optimization problem. To speed up, Furnival and Wilson (1974) introduced a well-known branch-and-bound algorithm with an efficient updating strategy for LMs, which was later implemented by R packages such as the **leaps** (Lumley and Miller 2017) and the **bestglm** (McLeod and Xu 2010). Hofmann, Gatu, Kontoghiorghes, Colubi, and Zeileis (2020) proposed an exact variable-subset selection algorithm for linear regression and implemented it in R package **lmSubsets**. Yet for GLMs, a simple exhaustive screen is undertaken in **bestglm**. When the exhaustive screening is not feasible for GLMs, fast approximating approaches have been proposed based on a genetic algorithm. For instance, **kofnGA** (Wolters 2015) implemented a genetic algorithm to search for a best subset of a pre-specified model size k, while **glmuti** (Calcagno and de Mazancourt 2010) implemented a genetic algorithm to automatically select the best model for GLMs with no more than 32 covariates. These packages can only deal with dozens of predictors but not high-dimensional data arising in modern statistics. Recently, Bertsimas, King, and Mazumder (2016) proposed a mixed integer optimization approach to find feasible best subset solutions for LMs with relatively larger p, which relies on certain third-party integer optimization solvers. Alternatively, regularization strategy is widely used to transform the subset selection problem into computational feasible problem. For example, **glmnet** (Friedman, Hastie, and Tibshirani 2010; Simon, Friedman, Hastie, and Tibshirani 2011) implemented a coordinate descent algorithm to solve the LASSO problem, which is a convex relaxation by replacing the cardinality constraint in best subset selection problem by the L_1 norm.

In this paper, we consider a primal-dual active set (PDAS) approach to exactly solve the best subset selection problem for sparse LM, GLM and CoxPH models. The PDAS algorithm for linear least squares problems was first introduced by Ito and Kunisch (2013) and later discussed by Jiao, Jin, and Lu (2015), Huang, Jiao, Liu, and Lu (2017) and Ghilli and Kunisch (2017). It utilizes an active set updating strategy and fits the sub-models through use of complementary primal and dual variables. We generalize the PDAS algorithm for general convex loss functions with the best subset constraint, and further extend it to support both sequential and golden section search strategies for optimal k determination. We develop a new package **BeSS** (best subset selection, Wen, Zhang, Quan, and Wang 2020) in the R programming system (R Core Team 2020) with C++ implementation of PDAS algorithms and memory optimized for sparse matrix output. This package is publicly available from the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/package=BeSS`. We demonstrate through enormous datasets that **BeSS** is efficient and stable for high dimensional data, and may solve best subset problems with n in 1000s and p in 10000s in just seconds on a single personal computer.

The article is organized as follows. In Section 2, we provide a general primal-dual formulation for the best subset problem that includes linear, logistic and CoxPH models as special cases. Section 3 presents the PDAS algorithms and related technical details. Numerical experiments based on enormous simulation and real datasets are conducted in Section 4. We conclude with a short discussion in Section 5.

2. Primal-dual formulation

The best subset selection problem with the subset size k is given by the following optimization problem:

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^p} l(\boldsymbol{\beta}) \quad \text{s.t.} \quad \|\boldsymbol{\beta}\|_0 = k, \tag{1}$$

where $l(\boldsymbol{\beta})$ is a convex loss function of the model parameters $\boldsymbol{\beta} \in \mathbb{R}^p$ and k is a positive integer. The L_0 norm $\|\boldsymbol{\beta}\|_0 = \sum_{j=1}^p |\beta_j|_0 = \sum_{j=1}^p 1_{\beta_j\neq 0}$ counts the number of nonzeros in $\boldsymbol{\beta}$.

It is known that the problem (1) admits non-unique local optimal solutions, among which the coordinate-wise minimizers possess promising properties. For a coordinate-wise minimizer $\boldsymbol{\beta}^\diamond$, denote the vectors of gradient and Hessian diagonal by

$$\mathbf{g}^\diamond = \nabla l(\boldsymbol{\beta}^\diamond), \quad \mathbf{h}^\diamond = \operatorname{diag}(\nabla^2 l(\boldsymbol{\beta}^\diamond)), \tag{2}$$

respectively. For each coordinate $j = 1, \dots, p$, write $l_j(t) = l(\beta_1^\diamond, \dots, \beta_{j-1}^\diamond, t, \beta_{j+1}^\diamond, \dots, \beta_p^\diamond)$ while fixing the other coordinates. Then the local quadratic approximation of $l_j(t)$ around $\beta_j^\diamond$ is given by

$$\begin{aligned} l_j^Q(t) &= l_j(\beta_j^\diamond) + g_j^\diamond(t-\beta_j^\diamond) + \tfrac{1}{2}h_j^\diamond(t-\beta_j^\diamond)^2 \\ &= \frac{1}{2}h_j^\diamond\left(t - \beta_j^\diamond + g_j^\diamond/h_j^\diamond\right)^2 + l_j(\beta_j^\diamond) - \frac{1}{2}[g_j^\diamond]^2/h_j^\diamond \\ &= \frac{1}{2}h_j^\diamond\left(t - (\beta_j^\diamond + \gamma_j^\diamond)\right)^2 + l_j(\beta_j^\diamond) - \frac{1}{2}[g_j^\diamond]^2/h_j^\diamond, \end{aligned} \tag{3}$$

which gives rise of an important quantity $\gamma_j^\diamond$ of the following scaled gradient form

$$\gamma_j^\diamond = -g_j^\diamond/h_j^\diamond. \tag{4}$$

Minimizing the objective function $l_j^Q(t)$ yields $t_j^* = \beta_j^\diamond + \gamma_j^\diamond$ for $j = 1, \dots, p$.

The constraint in (1) says that there are $p-k$ components of $\{t_j^*, j = 1, \dots, p\}$ that would be enforced to be zero. To determine which $p-k$ components, we consider the sacrifices of $l_j^Q(t)$ when switching each t_j^* from $\beta_j^\diamond + \gamma_j^\diamond$ to 0, which are given by

$$\Delta_j^\diamond = \frac{1}{2}h_j^\diamond(\beta_j^\diamond + \gamma_j^\diamond)^2, \quad j = 1, \dots, p. \tag{5}$$

Among all the candidates, we may enforce those t_j^*'s to zero if they contribute the *least total sacrifice* to the overall loss. To realize this, let $\Delta_{[1]}^\diamond \geq \cdots \geq \Delta_{[p]}^\diamond$ denote the decreasing rearrangement of $\Delta_j^\diamond$ for $j = 1, \dots, p$, then truncate the ordered sacrifice vector at position k. Combining the analytical result by (3), we obtain that

$$\beta_j^\diamond = \begin{cases} \beta_j^\diamond + \gamma_j^\diamond, & \text{if } \Delta_j^\diamond \geq \Delta_{[k]}^\diamond \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

In (6), we treat $\boldsymbol{\beta}^\diamond = (\beta_1^\diamond, \dots, \beta_p^\diamond)$ as primal variables, $\boldsymbol{\gamma}^\diamond = (\gamma_1^\diamond, \dots, \gamma_p^\diamond)$ as dual variables, and $\boldsymbol{\Delta}^\diamond = (\Delta_1^\diamond, \dots, \Delta_p^\diamond)$ as reference sacrifices. Next we provide their explicit expressions for three important statistical models.

Case 1: Linear regression. Consider the LM $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$ with design matrix $\mathbf{X} \in \mathbb{R}^{n\times p}$ and i.i.d. errors. Here $\mathbf{X}$ and $\mathbf{y}$ are standardized such that the intercept term is removed from the model and each column of $\mathbf{X}$ has $\sqrt{n}$ norm.
Take the loss function $l(\boldsymbol{\beta}) = \frac{1}{2n}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2$. For $j = 1, \dots, p$, it is easy to obtain

$$g_j^\diamond = \frac{1}{n}\mathbf{X}_{(j)}^\top(\mathbf{X}\boldsymbol{\beta} - \mathbf{y}), \quad h_j^\diamond = 1, \tag{7}$$

where $\mathbf{X}_{(j)}$ denotes the jth column of $\mathbf{X}$, so

$$\gamma_j^\diamond = \frac{1}{n}\mathbf{X}_{(j)}^\top(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}), \quad \Delta_j^\diamond = \frac{1}{2}(\beta_j^\diamond + \gamma_j^\diamond)^2. \tag{8}$$

Case 2: Logistic regression. Consider the GLM

$$\log(p(\mathbf{x})/(1 - p(\mathbf{x}))) = \beta_0 + \mathbf{x}^\top\boldsymbol{\beta}, \quad \mathbf{x} \in \mathbb{R}^p$$

with $p(\mathbf{x}) = \mathbb{P}(Y = 1|\mathbf{x})$. Given the data $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ with binary responses $y_i \in \{0, 1\}$, the negative log-likelihood function is given by

$$l(\beta_0, \boldsymbol{\beta}) = -\sum_{i=1}^n \Big\{y_i(\beta_0 + \mathbf{x}_i^\top\boldsymbol{\beta}) - \log(1 + \exp(\beta_0 + \mathbf{x}_i^\top\boldsymbol{\beta}))\Big\}. \tag{9}$$

We give only the primal-dual quantities for $\boldsymbol{\beta} \in \mathbb{R}^p$ according to the L_0 constraint in (1), while leaving β_0 to be estimated by unconstrained maximum likelihood method. For $j = 1, \dots, p$,

$$g_j^\diamond = -\sum_{i=1}^n x_{ij}(y_i - p_i^\diamond), \quad h_j^\diamond = \sum_{i=1}^n x_{ij}^2 p_i^\diamond(1 - p_i^\diamond) \tag{10}$$

where $p_i^\diamond = \exp(\beta_0 + \mathbf{x}_i^\top\boldsymbol{\beta}^\diamond)/(1 + \exp(\beta_0 + \mathbf{x}_i^\top\boldsymbol{\beta}^\diamond))$ denotes the i-th predicted probability. Then,

$$\gamma_j^\diamond = \frac{\sum_{i=1}^n x_{ij}(y_i - p_i^\diamond)}{\sum_{i=1}^n x_{ij}^2 p_i^\diamond(1 - p_i^\diamond)}, \quad \Delta_j^\diamond = \frac{1}{2}\sum_{i=1}^n x_{ij}^2 p_i^\diamond(1 - p_i^\diamond)(\beta_j^\diamond + \gamma_j^\diamond)^2. \tag{11}$$

Case 3: CoxPH regression. Consider the CoxPH model

$$\lambda(t|\mathbf{x}) = \lambda_0(t)\exp(\mathbf{x}^\top\boldsymbol{\beta}), \quad \mathbf{x} \in \mathbb{R}^p$$

with an unspecified baseline hazard $\lambda_0(t)$. Given the data $\{(T_i, \delta_i, \mathbf{x}_i) : i = 1, \dots, n\}$ with observations of survival time T_i and censoring indicator δ_i, by the method of partial likelihood (Cox 1972), the parameter $\boldsymbol{\beta}$ can be obtained by minimizing the following convex loss

$$l(\boldsymbol{\beta}) = -\sum_{i:\delta_i=1}\left(\mathbf{x}_i^\top\boldsymbol{\beta} - \log\left(\sum_{i':T_{i'}\geq T_i}\exp(\mathbf{x}_{i'}^\top\beta)\right)\right). \tag{12}$$

By writing $\omega_{i,i'}^\diamond = \exp(\mathbf{x}_{i'}^\top\boldsymbol{\beta}^\diamond)/\sum_{i':T_{i'}\geq T_i}\exp(\mathbf{x}_{i'}^\top\boldsymbol{\beta}^\diamond)$, it can be verified that

$$g_j^\diamond = -\sum_{i:\delta_i=1}\left(x_{ij} - \sum_{i':T_{i'}\geq T_i}\omega_{i,i'}^\diamond x_{i'j}\right) \tag{13}$$

$$h_j^\diamond = \sum_{i:\delta_i=1}\sum_{i':T_{i'}\geq T_i}\omega_{i,i'}^\diamond\left(x_{i'j} - \sum_{i':T_{i'}\geq T_i}\omega_{i,i'}^\diamond x_{i'j}\right)^2 \tag{14}$$

so that $\gamma_j^\diamond = -g_j^\diamond / h_j^\diamond$ and $\Delta_j^\diamond = \frac{1}{2} h_j^\diamond (\beta_j^\diamond + \gamma_j^\diamond)^2$ for $j = 1, \dots, p$.

3. Active set algorithm

For the best subset problem (1), define the active set $\mathcal{A} = \{j : \beta_j \neq 0\}$ with cardinality k and the inactive set $\mathcal{I} = \{j : \beta_j = 0\}$ with cardinality $p - k$. For the coordinate-wise minimizer $\boldsymbol{\beta}^\diamond$ satisfying (6), we have that

(C1) $\beta_j^\diamond = 0$ when $j \in \mathcal{I}$;

(C2) $\gamma_j^\diamond = 0$ when $j \in \mathcal{A}$;

(C3) $\Delta_j^\diamond \geq \Delta_{j'}^\diamond$ whenever $j \in \mathcal{A}$ and $j' \in \mathcal{I}$.

By (C1) and (C2), the primal variables $\beta_j^\diamond$'s and the dual variables $\gamma_j^\diamond$'s have complementary supports. (C3) can be viewed as a local stationary condition. These three conditions lay the foundation for the primal-dual active set algorithm we develop in this section.

Let $\mathcal{A}$ be a candidate active set. By (C1), we may estimate the k-nonzero primal variables by standard convex optimization:

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}_{\mathcal{I}} = \mathbf{0}}{\arg\min}\, l(\boldsymbol{\beta}), \quad \text{where } \mathcal{I} = \mathcal{A}^c. \tag{15}$$

Given $\hat{\boldsymbol{\beta}}$, the $\mathbf{g}$ and $\mathbf{h}$ vectors (2) can be computed, with their explicit expressions derived for linear, logistic and CoxPH models in the previous section. The $\boldsymbol{\gamma}$ and $\boldsymbol{\Delta}$ vectors are readily obtainable by (4), (5) and (C2). Then we may check if (C3) is satisfied; otherwise, update the active and inactive sets by

$$\mathcal{A} \leftarrow \left\{ j : \Delta_j \geq \Delta_{[k]} \right\}, \qquad \mathcal{I} \leftarrow \left\{ j : \Delta_j < \Delta_{[k]} \right\}. \tag{16}$$

This leads to the iterative algorithm shown in Algorithm 1.

Algorithm 1 Primal-dual active set (PDAS) algorithm.

```
1. Specify the cardinality $k$ of the active set and the maximum number of iterations $m_{\max}$.
   Initialize $\mathcal{A}$ to be a random $k$-subset of $\{1, \dots, p\}$ and $\mathcal{I} = \mathcal{A}^c$.

2. For $m = 1, 2, \dots, m_{\max}$, do
    (2.a) Estimate $\hat{\boldsymbol{\beta}}$ by (15).
    (2.b) Compute $\mathbf{g}, \mathbf{h}, \boldsymbol{\gamma}, \boldsymbol{\Delta}$.
    (2.c) Update $\mathcal{A}, \mathcal{I}$ by (16).
    (2.d) If $A$ is invariant, stop.

3. Output $\{\mathcal{A}, \hat{\boldsymbol{\beta}}, \boldsymbol{\Delta}\}$.
```

Algorithm 2 Sequential primal-dual active set (SPDAS) algorithm.

```
1. Specify the maximum size $k_{\max}$ of the active set, and initialize $\mathcal{A}^0 = \emptyset$.
2. For $k = 1, 2, \dots, k_{\max}$, do
       Run **PDAS** with initial $\mathcal{A}^{k-1} \cup \{j \in \mathcal{I}^{k-1} : j \in \arg\max \Delta_j^{k-1}\}$. Denote the output
       by $\{\mathcal{A}^k, \boldsymbol{\beta}^k, \boldsymbol{\Delta}^k\}$.
3. Output the optimal choice $\{\mathcal{A}^*, \boldsymbol{\beta}^*, \boldsymbol{\Delta}^*\}$ that attains the minimum AIC, BIC or EBIC.
```

Remark 1. *The proposed PDAS algorithm is based on the primal-dual active set strategy first developed by Ito and Kunisch (2013), but different from their original algorithm in two main aspects. First, our PDAS algorithm is derived from the quadratic argument (3) and it involves the second-order partial derivatives (i.e., Hessian diagonal* $\mathbf{h}$*). Second, our algorithm extends the original linear model setting to the general setting with convex loss functions.*

3.1. Determination of optimal k

The subset size k is usually unknown in practice, thus one has to determine it in a data-driven manner. A heuristic way is using the cross-validation technique to achieve the best prediction performance. Yet it is time consuming to conduct the cross-validation method especially for high-dimensional data. An alternative way is to run the PDAS algorithm from small to large k values, then identify an optimal choice according to some criteria, e.g., Akaike information criterion (Akaike 1974, AIC) and Bayesian information criterion (Schwarz 1978, BIC) and extended BIC (Chen and Chen 2008, 2012, EBIC) for small-n-large-p scenarios. This leads to the sequential PDAS algorithm shown in Algorithm 2.

To alleviate the computational burden of determining k as in SPDAS, here we provide an alternative method based on the golden section search algorithm. We begin by plotting the loss function $l(\boldsymbol{\beta})$ as a function of k for a simulated data from linear model with standard Gaussian error. The true coefficient $\boldsymbol{\beta} = (3, 1.5, 0, 0, -2, 0, 0, 0, -1, 0, \dots, 0)$ and the design matrix $\mathbf{X}$ is generated as in Section 4.1 with $\rho = 0.2$. From Figure 1, it can be seen that the slope of the loss plot goes from steep to flat and there is an 'elbow' exists near the true number of active set, i.e., $k = 4$.

The solution path for the same data is presented at the bottom of Figure 1 for a better visualization on the relationship between loss function and coefficient estimation. When a true active predictor is included in the model, the loss function drops dramatically and the predictors already in the model adjust their estimates to be close to the true values. When all the active predictors are included in the model, their estimates would not change much as k becomes larger.

Motivated by this interesting phenomenon, we develop a search algorithm based on the golden section method to determine the location of such an *elbow* in the loss function. In this way, we can avoid to run the PDAS algorithm extensively for a whole sequential list. The golden section primal-dual active set (GPDAS) algorithm is summarized in Algorithm 3.

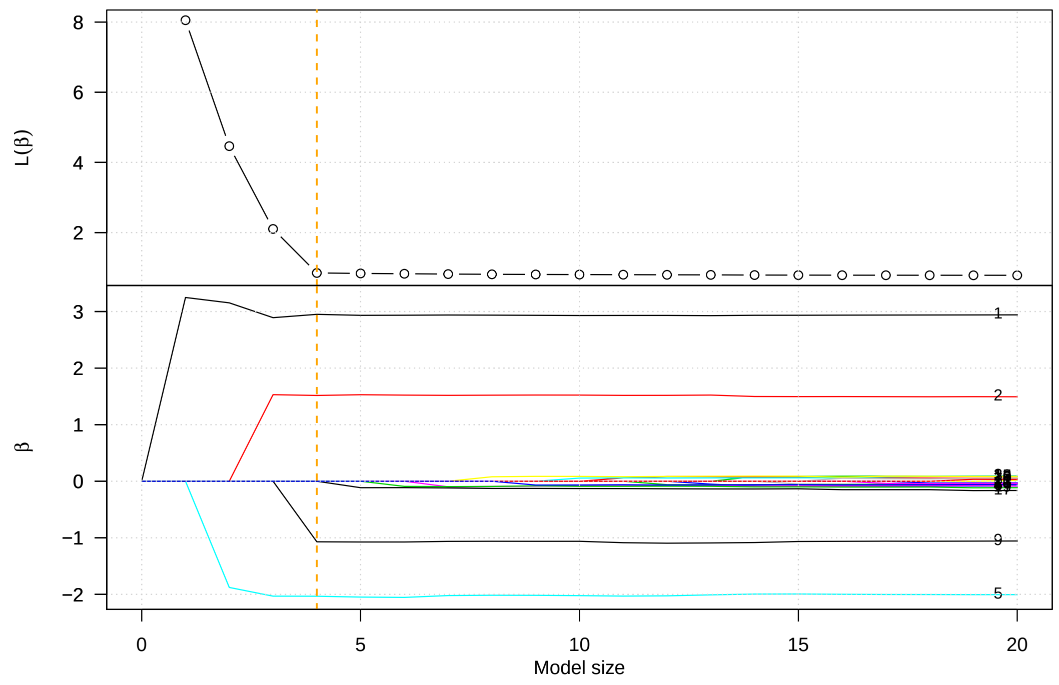

Figure 1: Plot of the loss function against the model complexity k and solution path for each coefficients. The orange vertical dash line indicates number of true nonzero coefficients.

3.2. Computational details

The proposed PDAS, SPDAS and SPDAS algorithms are much faster than existing methods reviewed in Section 1. For the exhaustive methods like **leaps** and **bestglm**, they essentially deal with $\sum_{k=1}^{k_{\max}} C(p,k)$ sub-models in order to search for the best subset with size no more than $k_{\max}$. It is infeasible even when $k_{\max}$ is moderate. That is why the greedy methods (e.g., **glmuti**) and the relaxed methods (e.g., **glmnet**) become popular. Our proposed algorithms belong to the greedy methods and their computational complexity is discussed below.

In general, consider one iteration in step (2) of the PDAS algorithm with a pre-specified k. Denote by N_l the computational complexity for solving $\boldsymbol{\beta}$ on the active set; and denote by N_g and N_h the computational complexity for calculating $\mathbf{g}$ and $\mathbf{h}$, respectively. The calculation of $\boldsymbol{\gamma}$ in steps (2.b)–(2.c) costs $O((p-k)\max(N_h, N_g))$, and the calculation of $\boldsymbol{\Delta}$ in steps (2.b)–(2.c) costs $O(pN_h)$. Then the overall cost of one iteration is $O(\max(N_l, pN_h, (p-k)N_g))$.

The total number of iterations of the PDAS algorithm could depend on the signal-to-noise ratio, the dimensionality p, and the sparsity level k. The algorithm may usually converge in finite steps (otherwise capped by $m_{\max}$). Denote by N_{P} the complexity for each run of the PDAS algorithm, then the total complexity of the SPDAS and GPDAS algorithms are $O(k_{\max} \times N_{\mathrm{P}})$ and $O(\log(k_{\max}) \times N_{\mathrm{P}})$, respectively.

Case 1: Linear regression. Since $\mathbf{h} = \mathbf{1}$, $N_h = 0$. The matrix vector product in the computation of $\mathbf{h}$ takes $O(n)$ flops. For the least squares problem on the active set, we use Cholesky factorization to obtain the estimate, which leads to $N_l = O(\max(nk^2, k^3))$. Thus the total cost of one iteration in step (2) is $O(\max(nk^2, k^3, n(p-k)))$, and the overall cost of the PDAS algorithm is the same since the number of iterations is often finite.

Algorithm 3 Golden section primal-dual active set (GPDAS) algorithm.

```
1. Specify the number of maximum iterations $m_{\max}$, the maximum size $k_{\max}$ of the active
   set and the tolerance $\eta \in (0,1)$. Initialize $k_L = 1$, and $k_R = k_{\max}$.
2. For $m = 1, 2, \ldots, m_{\max}$, do
   (2.a) Run PDAS with $k = k_L$ and initial $\mathcal{A}_L^{m-1} \cup \{j \in \mathcal{I}_L^{m-1} : j \in \arg\max(\Delta_L^{m-1})_j\}$.
         Output $\{\mathcal{A}_L^m, \boldsymbol{\beta}_L^m, \boldsymbol{\Delta}_L^m\}$.
   (2.b) Run PDAS with $k = k_R$ and initial $\mathcal{A}_R^{m-1} \cup \{j \in \mathcal{I}_R^{m-1} : j \in \arg\max(\Delta_R^{m-1})_j\}$.
         Output $\{\mathcal{A}_R^m, \boldsymbol{\beta}_R^m, \boldsymbol{\Delta}_R^m\}$.
   (2.c) Calculate $k_M = k_L + 0.618 \times (k_R - k_L)$. Run PDAS with $k = k_M$ and initial
         $\mathcal{A}_M^{m-1} \cup \{j \in \mathcal{I}_M^{m-1} : j \in \arg\max(\Delta_M^{m-1})_j\}$. Output $\{\mathcal{A}_M^m, \boldsymbol{\beta}_M^m, \boldsymbol{\Delta}_M^m\}$.
   (2.d) Determine whether $k_M$ is an elbow point:
         • Run PDAS with $k = k_M - 1$ and initial $\mathcal{A}_M^m$. Output $\{\mathcal{A}_{M-}^m, \boldsymbol{\beta}_{M-}^m, \boldsymbol{\Delta}_{M-}^m\}$.
         • Run PDAS with $k = k_M + 1$ and initial
           $citeal A_M^m$. Output $\{\mathcal{A}_{M+}^m, \boldsymbol{\beta}_{M+}^m, \boldsymbol{\Delta}_{M+}^m\}$.
         • If $|l(\boldsymbol{\beta}_M^m) - l(\boldsymbol{\beta}_{M_-}^m)| > \eta |l(\boldsymbol{\beta}_M^m)|$ and $|l(\boldsymbol{\beta}_M^m) - l(\boldsymbol{\beta}_{M+}^m)| < \eta |l(\boldsymbol{\beta}_M^m)|/2$, then
           stop and denote $k_M$ as an elbow point, otherwise go ahead.
   (2.e) Update $k_L, k_R$ and $\mathcal{A}_L^m, \mathcal{A}_R^m$:
         • If $|l(\boldsymbol{\beta}_M^m) - l(\boldsymbol{\beta}_L^m)| > \eta |l(\boldsymbol{\beta}_M^m)| > |l(\boldsymbol{\beta}_R^m) - l(\boldsymbol{\beta}_L^m)|$, then $k_R = k_M$, $\mathcal{A}_R^m = \mathcal{A}_M^m$;
         • If $\min\{|l(\boldsymbol{\beta}_M^m) - l(\boldsymbol{\beta}_L^m)|, |l(\boldsymbol{\beta}_R^m) - l(\boldsymbol{\beta}_L^m)|\} > \eta |l(\boldsymbol{\beta}_M^m)|$, then $k_L = k_M$, $\mathcal{A}_L^m =$
           $\mathcal{A}_M^m$;
         • Otherwise, $k_R = k_M, \mathcal{A}_R^m = \mathcal{A}_M^m$ and $k_L = 1, \mathcal{A}_L^m = \emptyset$.
   (2.f) If $k_L = k_R - 1$, then stop, otherwise $m = m + 1$.
3. Output $\{\mathcal{A}_M^m, \boldsymbol{\beta}_M^m, \boldsymbol{\Delta}_M^m\}$.
```

In particular, if the true coefficient vector is sparse with $k \ll p$ and $n = O(\log(p))$, the cost of the PDAS algorithm is $O(np)$, a linear time with respective to the size p. With an unknown k, we can choose an appropriate $k_{\max}$ value, e.g., $k_{\max} = n/\log(n)$, to speed up the SPDAS and GPDAS algorithms. Their costs become $O(n^2 p/\log(n))$ and $O(np\log(n/\log(n)))$, respectively. These rates are comparable with the sure independence screening procedure (Fan and Lv 2008) in handling ultrahigh-dimensional data. In fact, even if the true coefficient vector is not sparse, we could use a conjugate gradient (Golub and Van Loan 2012) algorithm with a preconditioning matrix to achieve a similar computational rate.

Case 2: Logistic regression. It costs $O(p)$ flop to compute the predicted probabilities p_i's. Thus $N_g = O(np)$ and $N_h = O(np)$. We use the iteratively reweighted least squares (IRLS) for parameter estimation on the active set. The complexity of each IRLS step is the same as that of the least squares, so $N_l = O(N_I \max(nk^2, k^3))$ with N_I denoting the finite number of IRLS iterations. The total cost of one iteration in step (2) is $O(\max(np^2, nk^2 N_I, k^3 N_I))$.

Case 3: CoxPH regression. It costs $O(np)$ flops to compute $\omega_{i,i'}$'s. Assume the censoring rate is c, then $N_g = O(n^3p(1-c))$ and $N_h = O(n^3p(1-c))$. Like the `coxph` command from the **survival** package (Therneau 2015), we adopt the standard Newton-Raphson algorithm for the maximum partial likelihood estimation on the active set. Its difficulty arises in the computation of the inverse of the Hessian matrix, which is full and dense. The Hessian matrix has k^2 entries and it requires $O(n^3k(1-c))$ flops for the computation of each entry. The matrix inversion costs $O(k^3)$ via Gauss-Jordan elimination or Cholesky decomposition. Hence, for each Newton-Raphson iteration, the updating equation requires $O(\max(n^3k^3(1-c), k^3))$ flops. We may speed up the algorithm by replacing the Hessian matrix with its diagonal, which reduces the computational complexity per updating to $O(\max(n^3k^2(1-c), k^3))$. Denote by N_{nr} the number of Newton-Raphson iterations, then $N_l = O(N_{nr}\max(n^3k^2(1-c), k^3))$ and the total cost of one iteration in step (2) is $O(\max(n^3p^2(1-c), n^3k^2(1-c)N_{nr}, k^3N_{nr}))$.

3.3. R package

We have implemented the active set algorithms described above into an R package called **BeSS** (best subset selection), which is publicly available from CRAN at `https://CRAN.R-project.org/package=BeSS`. The package is implemented in C++, using an **Rcpp** interface (Eddelbuettel and François 2011), with memory optimized using sparse matrix output and it can be called from R by a user-friendly interface.

The package contains two main functions, i.e., `bess.one` and `bess`, for solving the best subset selection problem with or without specification of k. In `bess`, two options are provided to determine the optimal k: one is based on the SPDAS algorithm with criteria including AIC, BIC and EBIC; the other is based on the GPDAS algorithm. The `plot` method for '`bess`' objects generates plots of loss functions for the best sub-models for each candidate k, together with solution paths for each predictor. We also include `predict` methods for '`bess`' and '`bess.one`' to make prediction on the new data.

4. Numerical examples

In this section we compare the performance of our new **BeSS** package to other well-known packages for best subset selection: **leaps**, **bestglm** and **glmulti**. We also include **glmnet** as an approximate subset selection method and use the default cross-validation method to determine an optimal tuning parameter. All parameters use the default values of the corresponding main functions in those packages unless otherwise stated. In presenting the results of **BeSS**, `bess.seq` represents `bess` with argument `method = "sequential"` and `bess.gs` represents `bess` with argument `method = "gsection"`, two different ways to determine the optimal parameter k. In `bess.seq`, we use AIC for examples with $n \geq p$ and EBIC for examples with $n < p$. We choose $k_{\max} = \min(n/2, p)$ for linear models and $k_{\max} = \min(n/\log(n), p)$ for logistic and CoxPH models.

All the R codes are demonstrated in Section 4.3. All computations are carried out on a 64-bit Intel machine with a single 3.30 GHz CPU and 4 GB of RAM.

4.1. Simulation data

We demonstrate the practical application of our new **BeSS** package on synthetical data under both low and high dimensional settings. For the low-dimensional data, **BeSS** has comparable performance with other state-of-the-art methods. For the high-dimensional data, while most state-of-the-art methods become incapable to deal with them, **BeSS** still performs fairly well. For an instance, **BeSS** is scalable enough to identify the best sub-model over all candidates efficiently in seconds or a few minutes when the dimension $p = 10000$.

We compare the performances of different methods in three aspects. The first aspect is the run time in seconds (Time). The second aspect is the selection performance in terms of true positive (TP) and false positive (FP) numbers, which are defined by the numbers of true relevant and true irrelevant variables among the selective predictors. The third aspect is the predictive performance on a held out test data of size 1000. For linear regression, we use the relative mean squares error (MSE) as defined by $\|\mathbf{X}\hat{\boldsymbol{\beta}} - \mathbf{X}\boldsymbol{\beta}^*\|_2/\|\mathbf{X}\boldsymbol{\beta}^*\|_2$. For logistic regression, we calculate the classification accuracy by the average number of observations being correctly classified. For CoxPH regression, we compute the median time on the test data, then derive the area under the receiver operator characteristic curve (i.e., AUC) using nearest neighbor estimation method as in Heagerty, Lumley, and Pepe (2000).

We generate the design matrix $\mathbf{X}$ and the underlying coefficients $\boldsymbol{\beta}$ as follows. The design matrix $\mathbf{X}$ is generated with $\mathbf{X}_{(j)} = \mathbf{Z}_j + 0.5 \times (\mathbf{Z}_{j-1} + \mathbf{Z}_{j+1}),\ j = 1, \ldots, p$, where $\mathbf{Z}_0 = \mathbf{0}, \mathbf{Z}_{p+1} = \mathbf{0}$ and $\{\mathbf{Z}_j, j = 1, \ldots, p\}$ were i.i.d. random samples drawn from standard Gaussian distribution and subsequently normalized to have $\sqrt{n}$ norm. The true coefficient $\boldsymbol{\beta}^*$ is a vector with q nonzero entries uniformly distributed in $[b, B]$, with b and B to be specified. In the simulation study, the sample size is fixed to be $n = 1000$. For each scenario, 100 replications are conducted .

Case 1: Linear regression. For each $\mathbf{X}$ and $\boldsymbol{\beta}^*$, we generate the response vector $\mathbf{y} = \mathbf{X}\boldsymbol{\beta}^* + \sigma\epsilon$, with $\epsilon \sim \mathcal{N}(0, 1)$. We set $b = 5\sigma\sqrt{2\log(p)/n},\ B = 100b$ and $\sigma = 3$. Different choices of (p, q) are taken to cover both the low-dimensional cases ($p = 20, 30, \text{or } 40,\ q = 4$) and the high-dimensional cases ($p = 100, 1000, \text{or } 10000,\ q = 40$). For `glmulti`, we only present the result for $p = 20$ and $p = 30$ since it can only deal with at most 32 predictors. Note that, due to the lack of possibility to set the random seeds, the results from `glmulti` are not exactly reproducible and they are subject to small variations. Since `leaps` and `bestglm` cannot deal with high-dimensional case, we only report the results of `glmnet`, `bess.seq` and `bess.gs`. The results are summarized in Table 1.

In the low-dimensional cases, the performances of all best subset selection methods are comparable in terms of prediction accuracy and selection consistency. However, the regularization method `glmnet` has much higher MSE and lower FP, which suggests that LASSO incurs bias in the coefficient estimation. In terms of computational times, both `bess.seq` and `bess.gs` have comparable performance with `glmnet`, which cost much less run times than the state-of-the-art methods. Unlike `leaps`, `bestglm` and `glmulti`, the run times of `bess.seq` and `bess.gs` remain fairly stable across different dimensionality.

In the high-dimensional cases, both `bess.seq` and `bess.gs` work quite well and they have similar performance in prediction and variable selection. Furthermore, their performances become better as p and q increase (from left to right in Table 1). On the other hand, `glmnet` has higher FP as p increases. In particular, when $p = 10000$ and only 40 nonzero coefficients are involved, the average TP equals 40 and the average FP is less than 3.06. In contrast, the average FP of `glmnet` increases to 30. As for the computational issues, both `bess.seq` and `bess.gs` seem to grow at a linear rate of p, but `bess.gs` offers speedups by factors of 2 up to 10 and more.

Low-dimensional		Method	$p = 20$	$p = 30$	$p = 40$
	Time	`leaps`	0.01 (0.01)	0.79 (0.23)	124.05 (46.68)
		`bestglm`	0.02 (0.01)	0.99 (0.26)	150.16 (53.93)
		`glmulti`	5.73 (1.17)	9.55 (2.19)	—
		`glmnet`	0.05 (0.01)	0.06 (0.01)	0.07 (0.01)
		`bess.seq`	0.13 (0.01)	0.17 (0.01)	0.22 (0.01)
		`bess.gs`	0.13 (0.01)	0.14 (0.02)	0.15 (0.01)
	MSE	`leaps`	2.02 (0.89)	2.16 (0.89)	2.47 (1.39)
	$(\times 10^{-2})$	`bestglm`	2.02 (0.89)	2.16 (0.89)	2.47 (1.39)
		`glmulti`	1.98 (0.90)	2.14 (0.87)	—
		`glmnet`	3.95 (1.49)	3.73 (1.29)	3.73 (2.38)
		`bess.seq`	2.01 (0.81)	2.17 (0.97)	2.45 (1.41)
		`bess.gs`	2.70 (5.00)	2.20 (4.57)	2.84 (3.57)
	TP	`leaps`	3.97 (0.17)	3.96 (0.24)	3.97 (0.17)
		`bestglm`	3.97 (0.17)	3.96 (0.24)	3.97 (0.17)
		`glmulti`	3.98 (0.14)	4.00 (0.00)	—
		`glmnet`	3.94 (0.24)	3.91 (0.32)	3.89 (0.35)
		`bess.seq`	3.96 (0.20)	3.91 (0.35)	3.84 (0.44)
		`bess.gs`	3.78 (0.48)	3.79 (0.50)	3.67 (0.53)
	FP	`leaps`	2.39 (1.56)	3.66 (2.17)	5.14 (2.45)
		`bestglm`	2.39 (1.56)	3.66 (2.17)	5.14 (2.45)
		`glmulti`	2.36 (1.70)	4.01 (2.14)	—
		`glmnet`	0.88 (0.87)	0.55 (0.76)	0.74 (1.01)
		`bess.seq`	4.19 (4.93)	5.84 (7.00)	6.75 (7.23)
		`bess.gs`	2.03 (4.18)	3.87 (6.61)	4.51 (7.64)
High-dimensional		Method	$p = 100$	$p = 1000$	$p = 10000$
	Time	`glmnet`	0.15 (0.01)	1.92 (0.09)	16.09 (0.14)
		`bess.seq`	1.09 (0.04)	80.74 (1.03)	150.62 (0.68)
		`bess.gs`	0.44 (0.09)	4.21 (0.40)	10.19 (0.97)
	MSE	`glmnet`	1.35 (0.15)	2.34 (0.28)	2.36 (0.19)
	$(\times 10^{-2})$	`bess.seq`	1.63 (0.37)	1.40 (0.78)	0.78 (0.39)
		`bess.gs`	1.38 (0.39)	0.96 (0.35)	0.95 (0.37)
	TP	`glmnet`	39.80 (0.45)	39.79 (0.46)	39.74 (0.50)
		`bess.seq`	35.52 (1.81)	38.45 (1.53)	39.40 (0.84)
		`bess.gs`	35.52 (2.34)	39.42 (0.88)	39.55 (0.78)
	FP	`glmnet`	15.81 (4.01)	15.74 (6.12)	38.27 (13.52)
		`bess.seq`	25.74 (10.41)	7.29 (8.97)	1.31 (4.00)
		`bess.gs`	28.83 (9.69)	1.31 (1.67)	3.16 (4.66)

Table 1: Simulation results for linear regression. Time stands for run time (CPU seconds), MSE stands for mean squared error, TP stands for true positive number and FP stands for false positive number. The number of true nonzero coefficients is $q = 4$ for low-dimensional cases and $q = 40$ for high-dimensional cases.

Case 2: Logistic regression. For each $\mathbf{x}$ and $\boldsymbol{\beta}^*$, the binary response is generated by $y = \text{Bernoulli}(\mathbb{P}(Y = 1))$, where $\mathbb{P}(Y = 1) = \exp(\mathbf{x}^\top \boldsymbol{\beta}^*)/(1 + \exp(\mathbf{x}^\top \boldsymbol{\beta}^*))$. The range of nonzero coefficients are set as $b = 10\sqrt{2\log(p)/n}$, $B = 5b$. Different choices of p are taken to cover both the low-dimensional cases ($p = 8, 10$, or 12) and the high-dimensional cases ($p = 100, 1000$, or 10000). The number of true nonzero coefficients is chosen to be $q = 4$ for low-dimensional cases and $q = 20$ for high-dimensional cases. Since `bestglm` is based on complete enumeration, it may be used for low-dimensional cases yet it becomes computationally infeasible for high dimensional cases.

The simulation results are summarized in Table 2. When p is small, both `bess.seq` and `bess.gs` have comparable performance with `bestglm`, `glmulti` and `glmnet`, but have considerably faster speed in computation than `bestglm` and `glmulti`. In the high-dimensional cases, we see that all three methods perform very well in terms of accuracy and TP. Yet both `bess.seq` and `bess.gs` have much smaller FP than `glmnet`. Among them, the run time for `bess.gs` is around a quarter of that for `bess.seq` and is similar to that for `glmnet`.

Case 3: CoxPH regression. For each $\mathbf{x}$ and $\boldsymbol{\beta}^*$, we generate data from the CoxPH model with hazard rate $\lambda(t|\mathbf{x}) = \exp(\mathbf{x}^\top \boldsymbol{\beta}^*)$. The ranges of nonzero coefficients are set same as those in logistic regression, i.e., $b = 10\sqrt{2\log(p)/n}$, $B = 5b$. Different choices of p are taken to cover both the low-dimensional cases ($p = 8, 10$, or 12) and the high-dimensional cases ($p = 100, 1000$, or 10000). The number of true nonzero coefficients is chosen to be $q = 4$ for low-dimensional cases and $q = 20$ for high-dimensional cases. Since `glmulti` cannot handle more than 32 predictors, we only report the low dimensional result for `glmulti`.

The simulation results are summarized in Table 3. Our findings about `bess.seq` and `bess.gs` are similar to those for the logistic regression.

4.2. Real data

We also evaluate the performance of the **BeSS** package in modeling several real data sets. Table 4 lists these instances and their descriptions. All datasets are saved as R data objects and available online with this publication.

We randomly split the data into a training set with two-thirds observations and a test set with remaining observations. Different best subset selection methods are used to identify the best sub-model. For each method, the run time in seconds (Time) and the size of selected model (MS) are recorded. We also include measurements of the predictive performance on test data according to the metrics as in Section 4.1. For reliable evaluation, the aforementioned procedure is replicated for 100 times.

The modeling results are displayed in Table 5. Again in low-dimensional cases, `bess` has comparable performance with the state-of-art algorithms (branch-and-bound algorithm for linear models and complete enumeration algorithm and genetic algorithm for GLMs). Besides, `bess.gs` has comparable run time with `glmnet` and is considerably faster than `bess.seq` especially in high-dimensional cases.

4.3. Code demonstration

We demonstrate how to use the package **BeSS** on a synthesis data as discussed in Section 3.1 and a real data in Section 4.2. Firstly, load **BeSS** and generate data with the `gen.data` function.

Low-dimensional		Method	$p = 8$	$p = 10$	$p = 12$
	Time	`bestglm`	1.39 (0.07)	5.92 (0.22)	24.78 (1.00)
		`glmulti`	1.37 (0.10)	8.59 (1.43)	12.90 (2.61)
		`glmnet`	0.69 (0.13)	0.87 (0.22)	0.99 (0.25)
		`bess.seq`	0.35 (0.14)	0.44 (0.14)	0.58 (0.19)
		`bess.gs`	0.31 (0.09)	0.432(0.13)	0.50 (0.13)
	Acc	`bestglm`	0.948 (0.011)	0.951 (0.011)	0.952 (0.013)
		`glmulti`	0.948 (0.011)	0.951 (0.011)	0.952 (0.013)
		`glmnet`	0.948 (0.011)	0.951 (0.011)	0.953 (0.013)
		`bess.seq`	0.948 (0.011)	0.951 (0.011)	0.952 (0.013)
		`bess.gs`	0.948 (0.012)	0.951 (0.011)	0.953 (0.013)
	TP	`bestglm`	3.99 (0.10)	3.99 (0.10)	3.99 (0.10)
		`glmulti`	3.99 (0.10)	3.99 (0.10)	3.99 (0.10)
		`glmnet`	4.00 (0.00)	4.00 (0.00)	4.00 (0.00)
		`bess.seq`	3.97 (0.22)	3.91 (0.32)	3.94 (0.24)
		`bess.gs`	3.9 (0.30)	3.87 (0.37)	3.88 (0.38)
	FP	`bestglm`	0.67 (0.79)	0.93 (1.12)	1.16 (1.15)
		`glmulti`	0.67 (0.79)	0.92 (1.11)	1.24 (1.18)
		`glmnet`	1.40 (0.93)	2.11 (1.24)	2.72 (1.30)
		`bess.seq`	1.49 (1.46)	2.10 (2.09)	2.50 (2.64)
		`bess.gs`	0.16 (0.49)	0.31 (0.91)	0.29 (0.88)
High-dimensional		Method	$p = 100$	$p = 1000$	$p = 10000$
	Time	`glmnet`	10.46 (1.83)	7.33 (0.55)	24.51 (0.54)
		`bess.seq`	83.81 (16.87)	63.44 (3.40)	75.00 (2.18)
		`bess.gs`	12.67 (3.85)	15.97 (2.91)	19.34 (4.29)
	Acc	`glmnet`	0.968 (0.006)	0.944 (0.009)	0.922 (0.012)
		`bess.seq`	0.963 (0.012)	0.974 (0.010)	0.979 (0.007)
		`bess.gs`	0.972 (0.008)	0.977 (0.007)	0.979 (0.008)
	TP	`glmnet`	19.97 (0.17)	19.97 (0.17)	19.78 (0.52)
		`bess.seq`	16.76 (2.10)	19.55 (0.93)	19.91 (0.35)
		`bess.gs`	18.79 (1.13)	19.85 (0.48)	19.88 (0.43)
	FP	`glmnet`	34.24 (4.61)	123.99 (19.45)	224.85 (38.56)
		`bess.seq`	4.63 (2.49)	1.39 (1.78)	0.55 (0.74)
		`bess.gs`	2.26 (2.06)	0.61 (0.70)	0.60 (1.02)

Table 2: Simulation results for logistic regression. Time stands for run time (CPU seconds), Acc stands for classification accuracy, TP stands for true positive number and FP stands for false positive number. The number of true nonzero coefficients is $q = 4$ for low-dimensional cases and $q = 20$ for high-dimensional cases.

```
R> library("BeSS")
R> set.seed(123)
R> Tbeta <- rep(0, 20)
R> Tbeta[c(1, 2, 5, 9)] <- c(3, 1.5, -2, -1)
R> data <- gen.data(n = 200, p = 20, family = "gaussian", beta = Tbeta,
+    rho = 0.2, sigma = 1)
```

Low-dimensional		Method	$p = 8$	$p = 10$	$p = 12$
	Time	`glmulti`	1.80 (0.06)	11.47 (1.82)	16.91 (3.40)
		`glmnet`	1.82 (0.43)	2.02 (0.41)	2.23 (0.51)
		`bess.seq`	0.39 (0.15)	0.45 (0.15)	0.55 (0.19)
		`bess.gs`	0.35 (0.12)	0.47 (0.14)	0.57 (0.16)
	AUC	`glmulti`	0.974 (0.010)	0.974 (0.009)	0.975 (0.012)
		`glmnet`	0.974 (0.010)	0.974 (0.009)	0.975 (0.012)
		`bess.seq`	0.974 (0.010)	0.974 (0.009)	0.975 (0.012)
		`bess.gs`	0.973 (0.010)	0.973 (0.009)	0.975 (0.012)
	TP	`glmulti`	4.00 (0.00)	4.00 (0.00)	4.00 (0.00)
		`glmnet`	4.00 (0.00)	4.00 (0.00)	4.00 (0.00)
		`bess.seq`	4.00 (0.00)	4.00 (0.00)	3.99 (0.10)
		`bess.gs`	3.94 (0.28)	3.91 (0.32)	3.98 (0.14)
	FP	`glmulti`	0.45 (0.73)	0.84 (0.97)	1.28 (1.22)
		`glmnet`	1.08 (0.92)	1.46 (1.05)	1.93 (1.21)
		`bess.seq`	1.30 (1.51)	1.91 (2.19)	2.54 (2.65)
		`bess.gs`	0.06 (0.28)	0.07 (0.26)	0.02 (0.14)
High-dimensional		Method	$p = 100$	$p = 1000$	$p = 10000$
	Time	`glmnet`	42.55 (4.48)	808.56 (200.51)	3233.67 (15690.86)
		`bess.seq`	130.76 (18.23)	1140.95 (141.00)	1729.05 (117.95)
		`bess.gs`	11.17 (6.12)	77.05 (117.66)	115.14 (12.30)
	AUC	`glmnet`	0.990 (0.007)	0.992 (0.006)	0.993 (0.005)
		`bess.seq`	0.990 (0.007)	0.992 (0.006)	0.993 (0.005)
		`bess.gs`	0.989 (0.008)	0.992 (0.006)	0.993 (0.005)
	TP	`glmnet`	20.00 (0.00)	20.00 (0.00)	20.00 (0.00)
		`bess.seq`	18.84 (1.22)	19.74 (0.65)	19.96 (0.28)
		`bess.gs`	17.73 (2.17)	19.98 (0.20)	20.00 (0.00)
	FP	`glmnet`	40.98 (4.03)	243.47 (23.93)	539.65 (32.37)
		`bess.seq`	10.96 (9.30)	1.45 (3.40)	0.22 (1.55)
		`bess.gs`	10.79 (10.71)	0.03 (0.30)	0.00 (0.00)

Table 3: Simulation results for CoxPH regression. Time stands for run time (CPU seconds), AUC stands for the integrated time-dependent area under the curve, TP stands for true positive number and FP stands for false positive number. The number of true nonzero coefficients is $q = 4$ for low-dimensional cases and $q = 20$ for high-dimensional cases.

We may call the `bess.one` function to solve the best subset selection problem with a specified cardinality. Then we can `print` or `summary` the '`bess.one`' object. While the `print` method allows users to obtain a brief summary of the fitted model, the `summary` method presents a much more detailed description.

```
R> fit.one <- bess.one(data$x, data$y, s = 4, family = "gaussian")
R> print(fit.one)
```

```
         Df          MSE          AIC          BIC         EBIC
  4.0000000   0.8494968 -24.6222136 -11.4289442  12.5369140
```

Dataset	n	p	Type	Data source
`prostate`	97	9	Continuous	R package **ElemStatLearn** (Halvorsen 2019)
`SAheart`	462	8	Binary	R package **ElemStatLearn**
`trim32`	120	18975	Continuous	Scheetz, Kim *et al.* (2006)
`leukemia`	72	3571	Binary	R package **spikeslab** (Ishwaran, Rao, and Kogalur 2013)
`gravier`	168	2905	Binary	`https://github.com/ramhiser/datamicroarray/wiki/Gravier-(2010)`
`er0`	609	22285	Survival	`https://www.ncbi.nlm.nih.gov/geo/`

Table 4: Description for the real data sets. Here n denotes the number of observations, p denotes the number of predictors, and "Type" denotes the type of response.

```
R> summary(fit.one)

----------------------------------------------------------------------
    Primal-dual active algorithm with maximum iteration being 15

    Best model with k = 4 includes predictors:

       X1         X2         X5         X9 
 2.950316  1.516192 -2.033013 -1.070717 

    log-likelihood:    16.31111
    deviance:         -32.62221
    AIC:              -24.62221
    BIC:              -11.42894
    EBIC:              12.53691
----------------------------------------------------------------------
```

The estimated coefficients of the fitted model can be extracted by using the `coef` function, which provides a sparse output with the control of argument `sparse = TRUE`. It is recommended to output a non-sparse vector when `bess.one` is used, and to output a sparse matrix when `bess` is used.

```
R> coef(fit.one, sparse = FALSE)

(intercept)          X1          X2          X3          X4          X5 
-0.08759837  2.95031623  1.51619150  0.00000000  0.00000000 -2.03301335 
         X6          X7          X8          X9         X10         X11 
 0.00000000  0.00000000  0.00000000 -1.07071719  0.00000000  0.00000000 
        X12         X13         X14         X15         X16         X17 
 0.00000000  0.00000000  0.00000000  0.00000000  0.00000000  0.00000000 
        X18         X19         X20 
 0.00000000  0.00000000  0.00000000 
```

Data	Method	`leaps`	`bestglm`	`glmulti`	`glmnet`	`bess.seq`	`bess.gs`
`prostate`	Time	0.00 (0.00)	0.01 (0.01)	0.43 (0.04)	0.05 (0.01)	0.10 (0.01)	0.11 (0.01)
	PE	0.60 (0.13)	0.60 (0.13)	0.60 (0.13)	0.68 (0.17)	0.60 (0.13)	0.58 (0.13)
	MS	4.28 (1.28)	4.24 (1.26)	4.24 (1.26)	3.34 (1.00)	4.25 (1.31)	6.20 (0.75)
`SAheart`	Time	—	1.14 (0.03)	2.66 (0.39)	0.13 (0.01)	0.15 (0.03)	0.17 (0.06)
	Acc	—	0.72 (0.03)	0.72 (0.03)	0.71 (0.04)	0.72 (0.03)	0.72 (0.03)
	MS	—	5.56 (0.77)	5.56 (0.77)	4.70 (0.86)	5.56 (0.81)	6.33 (1.06)
`trim32`	Time	—	—	—	3.03 (0.10)	1.23 (0.08)	0.68 (0.08)
	PE	—	—	—	0.01 (0.01)	0.01 (0.01)	0.01 (0.00)
	MS	—	—	—	24.55 (12.06)	1.62 (0.60)	8.28 (3.27)
`leukemia`	Time	—	—	—	0.45 (0.02)	1.05 (0.64)	0.68 (0.29)
	Acc	—	—	—	0.92 (0.05)	0.89 (0.07)	0.90 (0.06)
	MS	—	—	—	11.58 (4.23)	1.55 (0.76)	2.00 (0.00)
`gravier`	Time	—	—	—	0.91 (0.03)	3.13 (2.21)	2.52 (2.86)
	Acc	—	—	—	0.69 (0.06)	0.71 (0.06)	0.71 (0.06)
	MS	—	—	—	11.31 (7.01)	8.97 (1.09)	10.66 (2.39)
`er0`	Time	—	—	—	155.42 (19.90)	115.57 (53.23)	39.88 (15.49)
	AUC	—	—	—	0.52 (0.04)	0.53 (0.05)	0.60 (0.04)
	MS	—	—	—	3.23 (6.85)	1.04 (0.20)	58.97 (6.85)

Table 5: Results for the real data sets. Time stands for run time (CPU seconds), MS stands for the size of selected model. PE stands for mean prediction error in linear model; Acc stands for classification accuracy in logistic regression model; AUC stands for the integrated time-dependent area under the curve in CoxPH regression model.

To make prediction on new data, the `predict` function can be used as follows.

```
R> pred.one <- predict(fit.one, newdata = data$x)
```

To extract the selected best model, we provide the `lm`, `glm`, or `coxph` type of object named the `bestmodel` in the fitted '`bess.one`' object. Users could `print`, `summary` or `predict` this `bestmodel` object just like working with classical regression modeling. This would be helpful for statistical analysts who are familiar with `lm`, `glm`, or `coxph` functions.

```
R> bm.one <- fit.one$bestmodel
R> summary(bm.one)


Call:
lm(formula = ys ~ xbest, weights = weights)

Residuals:
    Min      1Q  Median      3Q     Max
-2.6612 -0.6411  0.0096  0.5791  3.2383

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.08760    0.06600  -1.327    0.186
xbestX1      2.95032    0.06713  43.950   <2e-16 ***
xbestX2      1.51619    0.06575  23.061   <2e-16 ***
xbestX5     -2.03301    0.06575 -30.921   <2e-16 ***
xbestX9     -1.07072    0.06311 -16.966   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.9334 on 195 degrees of freedom
Multiple R-squared:  0.9543,	Adjusted R-squared:  0.9534
F-statistic:  1019 on 4 and 195 DF,  p-value: < 2.2e-16
```

In practice when the best subset size is unknown, we have to determine the optimal choice of such sub-model size. The function `bess` provides two options: `method = "sequential"` corresponds to the SPDAS algorithm, and `method = "gsection"` corresponds to the GPDAS algorithm. Next we illustrate the usage of `bess` in the `trim32` data. We first load the data into the environment and show that it has 18975 variables, a much larger number compared with the sample size 120.

```
R> load("trim32.RData")
R> dim(X)

[1]   120 18975
```

Below is an example of running `bess` with argument `method = "sequential", epsilon = 0` and other argument being default values. We use the `summary` function to give a summary of the fitted '`bess`' object.

```
R> fit.seq <- bess(X, Y, method = "sequential", epsilon = 0)
R> summary(fit.seq)

-----------------------------------------------------------------------------
   Primal-dual active algorithm with tuning parameter determined by
   sequential method

   Best model determined by AIC includes 25 predictors with AIC = -890.9282

   Best model determined by BIC includes 25 predictors with BIC = -821.2409

   Best model determined by EBIC includes 2 predictors with EBIC = -561.2689
-----------------------------------------------------------------------------
```

As in the `bess.one`, the `bess` function outputs an '`lm`' type of object `bestmodel` associated with the selected best model. Here the `bestmodel` component outputs the largest fitted model since we did not use any early stopping rule as shown in the argument `epsilon = 0`.

```
R> bm.seq <- fit.seq$bestmodel
R> summary(bm.seq)

Call:
lm(formula = ys ~ xbest)

Residuals:
      Min        1Q    Median        3Q       Max
-0.039952 -0.012366 -0.001078  0.011401  0.075677

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)        5.618703   0.407769  13.779  < 2e-16 ***
xbest1368348_at   -0.089394   0.014563  -6.139 1.97e-08 ***
xbest1370558_a_at -0.122228   0.010712 -11.410  < 2e-16 ***
xbest1372548_at   -0.179410   0.012085 -14.846  < 2e-16 ***
xbest1377032_at   -0.062936   0.016733  -3.761 0.000294 ***
xbest1382223_at    0.497858   0.023655  21.047  < 2e-16 ***
xbest1388491_at    0.266606   0.021538  12.378  < 2e-16 ***
xbest1388657_at   -0.085292   0.015030  -5.675 1.53e-07 ***
xbest1389122_at   -0.101926   0.015317  -6.655 1.88e-09 ***
xbest1390269_at    0.106434   0.012130   8.774 7.40e-14 ***
xbest1378024_at   -0.123666   0.017614  -7.021 3.40e-10 ***
xbest1378552_at   -0.049578   0.010397  -4.768 6.77e-06 ***
xbest1379586_at   -0.066086   0.013526  -4.886 4.22e-06 ***
xbest1379772_at   -0.096651   0.010166  -9.507 2.05e-15 ***
xbest1379933_at    0.186271   0.015806  11.785  < 2e-16 ***
xbest1380696_at    0.028347   0.006882   4.119 8.19e-05 ***
xbest1380977_at    0.104704   0.018148   5.769 1.01e-07 ***
```

```
xbest1382392_at   -0.033764   0.005830  -5.791 9.21e-08 ***
xbest1384690_at   -0.083789   0.013985  -5.991 3.80e-08 ***
xbest1385015_at    0.131036   0.011803  11.102  < 2e-16 ***
xbest1385032_at    0.100631   0.012171   8.268 8.73e-13 ***
xbest1385395_at   -0.139164   0.010919 -12.745  < 2e-16 ***
xbest1385673_at    0.071119   0.011828   6.013 3.46e-08 ***
xbest1392605_at   -0.051400   0.008229  -6.246 1.21e-08 ***
xbest1394502_at    0.020363   0.006134   3.320 0.001283 **
xbest1398128_at   -0.084070   0.012728  -6.605 2.36e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.02241 on 94 degrees of freedom
Multiple R-squared:  0.981,	Adjusted R-squared:  0.976
F-statistic: 194.5 on 25 and 94 DF,  p-value: < 2.2e-16
```

Alternatively, we might use criteria like AIC to select the best model among a sequential list of candidate models. As shown above, the output of the `bess` function includes the AIC, BIC and EBIC values for best subset selection. Since the `trim32` data is high dimensional, we opt to use the EBIC criterion to determine the optimal model size. Then we run the `coef` function to extract the coefficients in the '`bess`' object and output the nonzero coefficients of the selected model.

```
R> K.opt.ebic <- which.min(fit.seq$EBIC)
R> coef(fit.seq)[, K.opt.ebic][which(coef(fit.seq)[, K.opt.ebic] != 0)]

 (intercept) 1382223_at 1388491_at
   0.8054785  0.5715478  0.3555834
```

We can also run the `predict` function for a given `newdata`. The argument `type` specifies which criteria is used to select the best fitted model.

```
R> pred.seq <- predict(fit.seq, newdata = data$x, type = "EBIC")
```

The `plot` routine provides the loss function plot for the sub-models with different k values, as well as solution paths for each predictor. It also adds a vertical dashed line to indicate the optimal k value as determined by EBIC. Figure 2 shows the result from the following R code.

```
R> plot(fit.seq, type = "both", breaks = TRUE, K = K.opt.ebic)
```

Next we call the function `bess` with argument `method = "gsection"` to perform the GPDAS algorithm. At each iteration, it outputs the split information.

```
R> fit.gs <- bess(X, Y, family = "gaussian", method = "gsection",
+    epsilon = 1e-2)

1-th iteration s.left:1 s.split:16 s.right:25
2-th iteration s.left:1 s.split:10 s.right:16
```

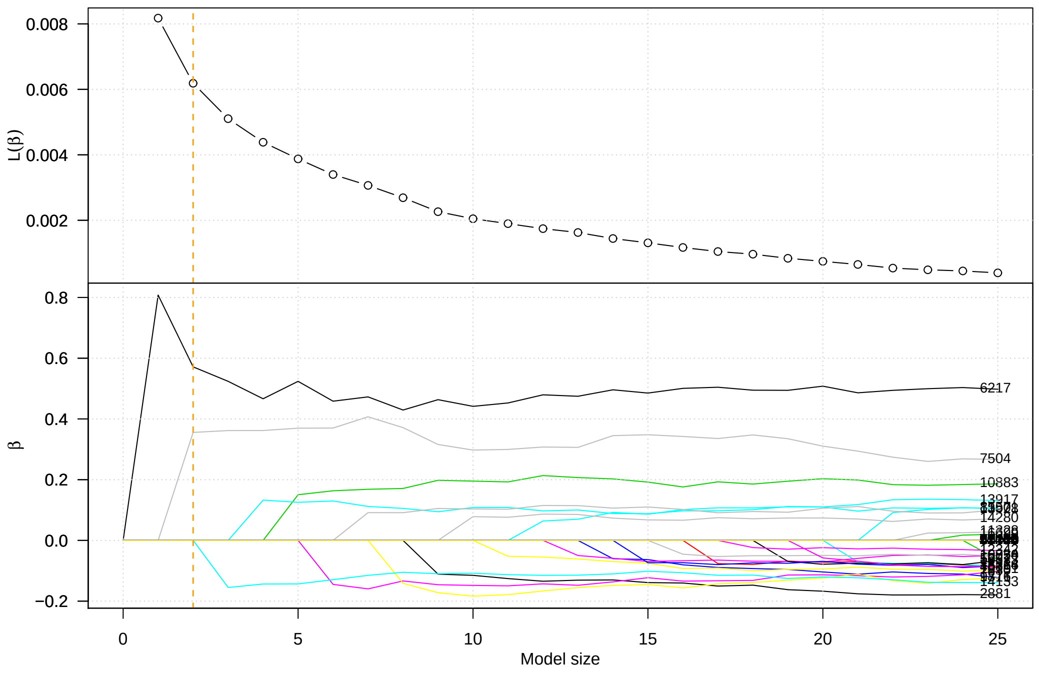

Figure 2: Best subset selection results for the `trim32` data with `bess.seq`. The optimal k value is determined by EBIC, which is indicated by a orange vertical dashed line.

```
3-th iteration s.left:1 s.split:7 s.right:10
4-th iteration s.left:1 s.split:5 s.right:7
5-th iteration s.left:5 s.split:6 s.right:7
```

From the above code, we know that the best selected model has 6 predictors and the algorithm ends at the 5th iteration. To show more information about the best selected model, we may extract `fit.gs$bestmodel` and present its summary information via the S3 method `summary`.

```
R> bm.gs <- fit.gs$bestmodel
R> summary(bm.gs)
```

```
Call:
lm(formula = ys ~ xbest)

Residuals:
      Min        1Q    Median        3Q       Max
-0.114598 -0.036829 -0.007365  0.041804  0.161688

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)        1.41219    0.50792   2.780 0.006363 **
xbest1368316_at   -0.16680    0.03479  -4.794 5.02e-06 ***
xbest1372248_at    0.22120    0.05579   3.965 0.000129 ***
```

```
xbest1373887_at    0.27947    0.05707   4.897 3.27e-06 ***
xbest1387160_at   -0.12456    0.02989  -4.168 6.05e-05 ***
xbest1389910_at    0.54459    0.07220   7.543 1.25e-11 ***
xbest1381978_a_at -0.16091    0.03454  -4.658 8.77e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.05926 on 113 degrees of freedom
Multiple R-squared:  0.8405,	Adjusted R-squared:  0.8321
F-statistic: 99.28 on 6 and 113 DF,  p-value: < 2.2e-16
```

Running the `coef` function directly on the '`bess`' object returns a sparse matrix as shown below. The last column corresponds to the best fitted coefficients.

```
R> beta <- coef(fit.gs, sparse = TRUE)
R> class(beta)

[1] "dgCMatrix"
attr(,"package")
[1] "Matrix"

R> beta[, ncol(beta)][which(beta[, ncol(beta)] != 0)]

 (intercept)   1368316_at   1372248_at   1373887_at   1387160_at
   1.4121869   -0.1668030    0.2211982    0.2794672   -0.1245576
  1389910_at 1381978_a_at
   0.5445936   -0.1609133
```

Based on the `fit.gs`, we can predict for the new data via the `predict` function as follows.

```
R> pred.gs <- predict(fit.gs, newdata = X)
```

5. Discussion

In this paper, we introduce the primal dual active set (PDAS) algorithm for solving the best subset selection problem under the general convex loss setting. The PDAS algorithm identifies the best sub-model with a pre-specified model size via a primal-dual formulation on feasible solutions. To determine the best sub-model over different model sizes, both sequential search and golden section search are proposed, i.e., SPDAS and GPDAS algorithms. We find that the GPDAS algorithm is especially efficient and accurate in selecting variables for high-dimensional and sparse data.

The proposed algorithms are implemented with C++ through the new **BeSS** package in the R statistical environment. Package **BeSS** provides R users with a new and flexible way to carry out best subset selection for sparse LM, GLM and CoxPH models. It allows us to identify the best sub-model efficiently (usually in seconds or a few minutes) even when the number of predictors is extremely large, say $p \approx 10000$, based on a standard personal computer. In both simulation and real data examples, it was shown that the **BeSS** package is highly efficient compared to other state-of-the-art methods.

Acknowledgments

We are grateful to the anonymous referees for valuable comments that lead to the improvement of the current paper. Wen is partially supported by NSFC (11801540), Natural Science Foundation of Guangdong (2017A030310572) and Fundamental Research Funds for the Central Universities (WK2040170015). Zhang is partially supported by Basic Research Seed Fund (201611159250) and Big Data Project Fund of The University of Hong Kong. Wang is partially supported by NSFC (11771462) and International Science & Technology Cooperation Program of Guang Dong(20163400042410001).

References

Akaike H (1974). "A New Look at the Statistical Model Identification." *IEEE Transactions on Automatic Control*, **19**(6), 716–723. doi:10.1109/tac.1974.1100705.

Bertsimas D, King A, Mazumder R (2016). "Best Subset Selection via a Modern Optimization Lens." *The Annals of Statistics*, **44**(2), 813–852. doi:10.1214/15-aos1388.

Calcagno V, de Mazancourt C (2010). "**glmulti**: An R Package for Easy Automated Model Selection with (Generalized) Linear Models." *Journal of Statistical Software*, **34**(12), 1–29. doi:10.18637/jss.v034.i12.

Chen J, Chen Z (2008). "Extended Bayesian Information Criteria for Model Selection with Large Model Spaces." *Biometrika*, **95**(3), 759–771.

Chen J, Chen Z (2012). "Extended BIC for Small-n-Large-P Sparse GLM." *Statistica Sinica*, **22**, 555–574. doi:10.5705/ss.2010.216.

Cox DR (1972). "Regression Models and Life-Tables." *Journal of the Royal Statistical Society B*, **34**(2), 187–220. doi:10.1111/j.2517-6161.1972.tb00899.x.

Eddelbuettel D, François R (2011). "**Rcpp**: Seamless R and C++ Integration." *Journal of Statistical Software*, **40**(8), 1–18. doi:10.18637/jss.v040.i08.

Fan J, Lv J (2008). "Sure Independence Screening for Ultrahigh Dimensional Feature Space." *Journal of the Royal Statistical Society B*, **70**(5), 849–911. doi:10.1111/j.1467-9868.2008.00674.x.

Friedman J, Hastie T, Tibshirani R (2010). "Regularization Paths for Generalized Linear Models via Coordinate Descent." *Journal of Statistical Software*, **33**(1), 1–22. doi:10.18637/jss.v033.i01.

Furnival GM, Wilson RW (1974). "Regressions by Leaps and Bounds." *Technometrics*, **16**(4), 499–511. doi:10.2307/1267601.

Ghilli D, Kunisch K (2017). "On the Monotone and Primal-Dual Active Set Schemes for ℓ_p-Type Problems, $p \in (0, 1]$." arXiv:1709.06506 [math.OC], URL https://arxiv.org/abs/1709.06506.

Golub GH, Van Loan CF (2012). *Matrix Computations*, volume 3. 4 edition. Johns Hopkins University Press.

Halvorsen KB (2019). ***ElemStatLearn**: Data Sets, Functions and Examples from the Book: "The Elements of Statistical Learning, Data Mining, Inference, and Prediction" by Trevor Hastie, Robert Tibshirani and Jerome Friedman.* R package version 2015.6.26.2, URL https://CRAN.R-project.org/src/contrib/Archive/ElemStatLearn/.

Heagerty PJ, Lumley T, Pepe MS (2000). "Time-Dependent ROC Curves for Censored Survival Data and a Diagnostic Marker." *Biometrics*, **56**(2), 337–344. doi:10.1111/j.0006-341x.2000.00337.x.

Hofmann M, Gatu C, Kontoghiorghes EJ, Colubi A, Zeileis A (2020). "**lmSubsets**: Exact Variable-Subset Selection in Linear Regression for R." *Journal of Statistical Software*, **93**(3), 1–21. doi:10.18637/jss.v093.i03.

Huang J, Jiao Y, Liu Y, Lu X (2017). "A Constructive Approach to High-Dimensional Regression." arXiv:1701.05128 [stat.CO], URL https://arxiv.org/abs/1701.05128.

Ishwaran H, Rao JS, Kogalur UB (2013). ***spikeslab**: Prediction and Variable Selection Using Spike and Slab Regression.* R package version 1.1.5, URL https://CRAN.R-project.org/package=spikeslab.

Ito K, Kunisch K (2013). "A Variational Approach to Sparsity Optimization Based on Lagrange Multiplier Theory." *Inverse Problems*, **30**(1), 015001.

Jiao Y, Jin B, Lu X (2015). "A Primal Dual Active Set with Continuation Algorithm for the ℓ^0-Regularized Optimization Problem." *Applied and Computational Harmonic Analysis*, **39**(3), 400–426. doi:10.1016/j.acha.2014.10.001.

Lumley T, Miller A (2017). ***leaps**: Regression Subset Selection.* R package version 3.0, URL https://CRAN.R-project.org/package=leaps.

McLeod AI, Xu C (2010). ***bestglm**: Best Subset GLM and Regression Utilities.* R package version 0.36, URL https://CRAN.R-project.org/package=bestglm.

R Core Team (2020). *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Scheetz TE, Kim KYA, *et al.* (2006). "Regulation of Gene Expression in the Mammalian Eye and Its Relevance to Eye Disease." *Proceedings of the National Academy of Sciences of the United States of America*, **103**(39), 14429–14434. doi:10.1073/pnas.0602562103.

Schwarz G (1978). "Estimating the Dimension of a Model." *The Annals of Statistics*, **6**(2), 461–464.

Simon N, Friedman J, Hastie T, Tibshirani R (2011). "Regularization Paths for Cox's Proportional Hazards Model via Coordinate Descent." *Journal of Statistical Software*, **39**(5), 1–13. doi:10.18637/jss.v039.c05.

Therneau TM (2015). ***survival**: A Package for Survival Analysis in S.* R package version 2.38, URL https://CRAN.R-project.org/package=survival.

Wen C, Zhang A, Quan S, Wang X (2020). ***BeSS**: Best Subset Selection for Sparse Generalized Linear Model and Cox Model.* R package version 1.0.9, URL https://CRAN.R-project.org/package=BeSS.

Wolters MA (2015). "A Genetic Algorithm for Selection of Fixed-Size Subsets with Application to Design Problems." *Journal of Statistical Software*, **68**(1), 1–18. doi:10.18637/jss.v068.c01.

Affiliation:

Canhong Wen, Xueqin Wang *(corresponding author)*
Department of Statistics and Finance, School of Management
University of Science and Technology of China
230026 Hefei, AH, China
E-mail: wench@ustc.edu.cn, wangxq20@ustc.edu.cn
and
Department of Statistics, School of Mathematics
Southern China Center for Statistical Science
Sun Yat-sen University
510275 Guangzhou, GD, China

Aijun Zhang
Department of Statistics and Actuarial Science
The University of Hong Kong
Hong Kong, China
E-mail: ajzhang@hku.hk

Shijie Quan
Department of Statistics, School of Mathematics
Southern China Center for Statistical Science
Sun Yat-sen University
510275 Guangzhou, GD, China

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
June 2020, Volume 94, Issue 4 *Submitted:* 2017-06-30
doi:10.18637/jss.v094.i04 *Accepted:* 2019-11-10