coxphMIC: An R Package for Sparse Estimation of Cox Proportional Hazards Models via Approximated Information Criteria

by Razieh Nabi and Xiaogang Su

Abstract In this paper, we describe an R package named **coxphMIC**, which implements the sparse estimation method for Cox proportional hazards models via approximated information criterion (Su et al., 2016). The developed methodology is named MIC which stands for "Minimizing approximated Information Criteria". A reparameterization step is introduced to enforce sparsity while at the same time keeping the objective function smooth. As a result, MIC is computationally fast with a superior performance in sparse estimation. Furthermore, the reparameterization tactic yields an additional advantage in terms of circumventing post-selection inference (Leeb and Pötscher, 2005). The MIC method and its R implementation are introduced and illustrated with the PBC data.

Introduction

Time to event (survival time) is often a primary outcome of interest in many research areas, especially in medical research such as time that takes to respond to a particular therapy, time to death, remission, or relapse. Survival times are typically right skewed and subject to censoring due to study termination, loss of follow ups, or withdrawals. Moreover, covariates may vary by time.

Cox Proportional Hazards (PH) model (Cox, 1972) is commonly used to model survival data. Given a typical survival data set that consists of $\{(T_i, \delta_i, \mathbf{z}_i) : i = 1, \ldots, n\}$, where T_i is the observed event time, δ_i is the 0-1 binary censoring indicator, and $\mathbf{z}_i \in \mathbb{R}^p$ is the covariate vector associated with the i-th subject, the Cox PH model formulates the hazard function $h(t|\mathbf{z_i})$ for the ith subject as

$$h(t|\mathbf{z_i}) = h_0(t)\ \exp(\boldsymbol{\beta}^T\ \mathbf{z_i}),$$

where $\mathbf{z_i} \in \mathbb{R}^p$ denotes the p-dimensional covariate vector associated with subject i, $\boldsymbol{\beta} = (\beta_j) \in \mathbb{R}^p$ is the unknown regression parameter vector, and $h_0(t)$ is the unspecified baseline hazard function. The vector of $\boldsymbol{\beta}$ can be estimated by maximizing the partial log-likelihood (Cox, 1975), which is given by

$$l(\boldsymbol{\beta}) \ = \ \sum_{i=1}^{n} \delta_i \left[\mathbf{z_i}^T \boldsymbol{\beta} - \log \sum_{i'=1}^{n} \left\{ I(T_{i'} \geq T_i)\ \exp(\mathbf{z_{i'}}^T \boldsymbol{\beta}) \right\} \right].$$

Let $\widehat{\boldsymbol{\beta}}$ denote the resultant maximum partial likelihood estimator (MPLE).

Since the true $\boldsymbol{\beta}$ is often sparse, we need to look for methods that identify the zero components in $\boldsymbol{\beta}$ and at the same time estimate the nonzero ones. Best subset selection (BSS) and regularization are among two major algorithms used in survival analyses for variable selection. Both are derived from a penalized partial likelihood. Let $\text{pen}(\boldsymbol{\beta})$ and λ denote the penalty function and penalty parameter, respectively. The general objective function in both of the techniques is as follows:

$$\min_{\boldsymbol{\beta}}\ \ -2l(\boldsymbol{\beta}) + \lambda \cdot \text{pen}(\boldsymbol{\beta}).$$

In BSS, the penalty function is set to $\text{pen}(\boldsymbol{\beta}) = \sum_{j=1}^{p} I\{\beta_j \neq 0\}$ (number of nonzero coefficients), and the penalty parameter is fixed as $\lambda = 2$ for AIC (Akaike, 1974) or $\lambda = \ln(n_0)$, where n_0 is the total number of uncensored failures, with a slight modification of BIC (Vollinsk and Raftery, 2000). In regularization, the penalty function is set to $\text{pen}(\boldsymbol{\beta}) = \sum_{j=1}^{p} |\beta_j|$, and the penalty parameter is not fixed and is appropriately chosen. The sparse estimation is reformulated into a continuous convex optimization problem. The optimization of the two techniques is a two-step process. In BSS, one needs to fit every model with the maximum partial likelihood method and then compare the fitted models according to an information criterion such as AIC (Akaike, 1974) or BIC (Schwarz, 1978). This makes the BSS infeasible for moderately large p. In regularization, one need to solve the objective function for every fixed positive value of λ to obtain a regularization path $\{\widetilde{\boldsymbol{\beta}}(\lambda) : \lambda > 0\}$, and then select the best λ according to an information criterion such as AIC or BIC along the path. Since such a search is only along the regularization path (a one-dimensional curve in $\mathbb{R}^p$), the search space is much reduced and hence, it may not perform as well as the estimator obtained with BSS, if AIC or BIC is used as the yardstick. Beside the computational burden, both methods face the post-selection inference challenge. A new technique is developed by Su et al. (2016) on the basis of Su (2015) for conducting sparse estimation of Cox PH models to help address the aforementioned deficiencies.

The MIC method

A new method, named MIC for "Minimizing approximated Information Criteria", is developed to conduct sparse estimation of Cox PH models. MIC borrows strength from both BSS and regularization. The main issue with BSS is the indicator function, $I(\beta \neq 0)$, involved in the ℓ_0 penalty function, leading to a discrete optimization problem. To overcome this difficulty, MIC proposes to approximate the indicator function by a continuous or smooth unit dent function. One reasonable approximation is the hyperbolic tangent function given by

$$w(\beta) \; = \; \tanh(a\beta^2) \; = \; \frac{\exp\left(a\beta^2\right) - \exp\left(-a\beta^2\right)}{\exp\left(a\beta^2\right) + \exp\left(-a\beta^2\right)},$$

where a is a nonnegative scale parameter that controls the sharpness of the approximation.

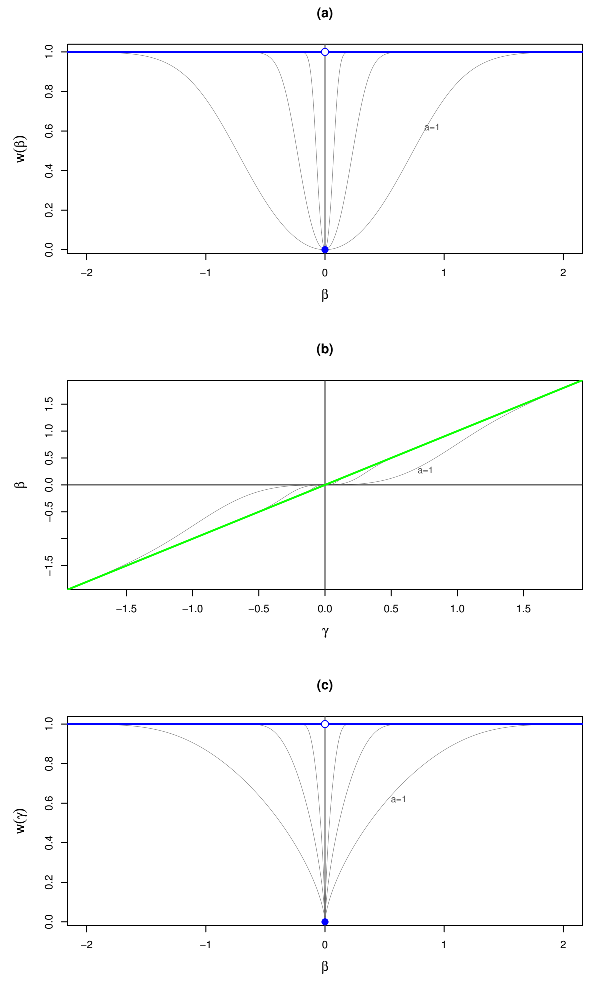

Figure 1: MIC penalty and the reparameterization step: (a) the hyperbolic tangent penalty $\tanh(a\beta^2)$ versus β; (b) $\beta = \gamma \tanh(a\gamma^2)$ versus γ; (c) $\tanh(a\gamma^2)$ as a penalty function of β. Three values of $a \in \{1, 10, 100\}$ are illustrated.

As shown in Figure 1(a), $w(\beta)$ provides a smooth approximation to the discrete function $I\{\beta \neq 0\}$. However, the curve does not have zero as a singular point. If we estimate $\boldsymbol{\beta}$ by minimizing $-2\,l(\boldsymbol{\beta}) + \ln(n_0) \sum_{j=1}^{p} w(\beta_j)$, we will not obtain sparse estimates. To enforce sparsity, MIC devises a reparameterization step. The reparameterization is based on the decomposition $\beta = \beta\, I\{\beta \neq 0\}$. Set $\gamma = \beta$ and approximate $I\{\beta \neq 0\}$ with $w(\gamma) = \tanh(a\gamma^2)$. This leads to a reparameterization of

$\beta = \gamma w(\gamma)$. The objective function in MIC is given by

$$Q_n(\boldsymbol{\beta}) \;=\; -2\,l(\mathbf{W}\gamma) + \lambda_0\,\mathrm{tr}(\mathbf{W}), \tag{1}$$

where the penalty parameter λ_0 is fixed as $\ln(n_0)$ for BIC (Vollinsk and Raftery, 2000) and matrix $\mathbf{W}$ is $p \times p$ diagonal with diagonal elements $w_j = w(\gamma_j)$ and hence trace $\mathrm{tr}(\mathbf{W}) = \sum_{j=1}^{p} \tanh(a\gamma_j^2)$. With this notation, it follows that $\boldsymbol{\beta} = \mathbf{W}\boldsymbol{\gamma}$.

The above reparameterization offers several important conveniences:

1. Sparsity now becomes achievable in estimating $\boldsymbol{\beta}$. The penalty $w(\gamma)$ as a function of $\beta = \gamma w(\gamma) = \gamma \tanh(\gamma)$ is a unit dent function that is smooth everywhere except at $\beta = 0$, as shown in Figure 1(c). This is a necessary condition to ensure sparsity as indicated by Fan and Li (2002). On this basis, the oracle properties of the MIC estimator $\widetilde{\boldsymbol{\beta}}$ obtained by minimizing $Q_n(\boldsymbol{\beta})$ in (1)

   $$\widetilde{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}}\, Q_n(\boldsymbol{\beta}) = \arg\min_{\boldsymbol{\beta}} -2\,l(\boldsymbol{\beta}) + \ln(n_0) \sum_{j=1}^{p} w(\gamma_j)$$

   can be established under regularity conditions. The asymptotic results entails $a_n = O(n)$. For this reason, we fix $a_n = n_0$, the number of non-censored failures. In practice, the empirical performance of MIC is large stable with respect to the choice of a, as demonstrated in Su (2015). Thus simply fixing a at a reasonably large value (say, $a \geq 10$) could do as well practically.

2. In terms of practical optimization, it is preferable to consider $\boldsymbol{\gamma}$ as the decision vector. Namely, we minimize $Q_n(\boldsymbol{\gamma})$ with respect to $\boldsymbol{\gamma}$ by treating it as a function of $\boldsymbol{\gamma}$. Let $\widetilde{\boldsymbol{\gamma}}$ be the resultant MIC estimator of $\boldsymbol{\gamma}$

   $$\widetilde{\boldsymbol{\gamma}} = \arg\min_{\boldsymbol{\gamma}}\, Q_n(\boldsymbol{\gamma}) = \arg\min_{\boldsymbol{\gamma}} -2\,l(\mathbf{W}\gamma) + \ln(n_0) \sum_{j=1}^{p} w(\gamma_j). \tag{2}$$

   One immediate advantage of doing so is that $Q_n(\boldsymbol{\gamma})$ is smooth in $\boldsymbol{\gamma}$ and hence many optimization routines can be applied directly. Since no selection of tuning parameters is involved, MIC is computationally efficient.

3. One consequence of post-selection inference is that no standard error formula is available for zero estimates of β_j. As depicted in Figure 1(b), β_j and γ_j have a one-to-one correspondence with $\beta = 0$ iff $\gamma = 0$. This motivates us to test $H_0: \;\beta_j = 0$ by equivalently testing $H_0: \;\gamma_j = 0$. The MIC estimator $\widetilde{\gamma}$ can be viewed as an M-estimator with smooth objective function $Q_n(\gamma)$ and hence standard arguments can be used to make inference.

Implementation in R

The R package **coxphMIC** implements MIC on the basis of R package `survival` (Therneau and Grambsch, 2000) and is hosted at CRAN. Type the following command in R console in order to install the package:

```
> install.packages("coxphMIC")
```

To summarize, MIC can be simply formulated as the following optimization problem

$$\min_{\boldsymbol{\gamma}} \;\; -2l(\mathbf{W}\boldsymbol{\gamma}) \,+\, \ln(n_0) \sum_{j=1}^{p} \tanh(n_0\gamma_j^2). \tag{3}$$

Owing to the non-convex nature, a global optimization method is helpful in solving (3). While other R routines (Mullen, 2014) are available, we have found using the `SANN` method combined with the `BFGS` method in R function `optim()` is fast and quite effective. The simulated annealing (SA) implemented by `SANN` helps locate a nearly minimum point globally. Then the quasi-Newton BFGS method makes sure that the algorithm stops at a critical point.

There are two functions included in the **coxphMIC** package: an internal function `LoglikPen()` that computes the partial log-likelihood and a wrapper function `coxphMIC()` that does the MIC sparse estimation. The function `coxphMIC()` has the following usage:

```
coxphMIC(formula = Surv(time, status) ~ ., data, method.beta0 = "MPLE",
    beta0 = NULL, theta0 = 1, method = "BIC", lambda0 = NULL, a0 = NULL,
    scale.x = TRUE, maxit.global = 300, maxit.local = 100,
    rounding.digits = 4, zero = sqrt(.Machine$double.eps),
```

Table 1: Comparison of computation time: CPU time (in seconds) averaged over three runs.

		Censoring	Method					
n	p	Rate	Full	Stepwise	MIC	LASSO	ALASSO	SCAD
200	10	25%	0.007	0.307	0.067	0.157	0.163	5.923
		40%	0.000	0.320	0.060	0.150	0.170	4.900
	50	25%	0.027	18.957	0.063	0.397	0.417	5.587
		40%	0.027	18.107	0.057	0.453	0.480	5.480
	100	25%	0.060	189.040	0.057	1.450	1.387	—
		40%	0.067	181.147	0.057	2.053	1.897	—
2000	10	25%	0.020	0.907	0.243	0.903	0.837	415.097
		40%	0.017	0.880	0.240	0.893	0.823	328.150
	50	25%	0.110	81.380	0.243	1.590	1.153	—
		40%	0.093	72.887	0.237	1.613	1.163	—
	100	25%	0.333	894.607	0.223	2.383	2.103	—
		40%	0.240	673.503	0.187	2.073	1.357	—

```
compute.se.gamma = TRUE, compute.se.beta = TRUE,
CI.gamma = TRUE, conf.level = 0.95,
details = FALSE)
```

We briefly explain some of the important options. The `formula` argument is a formula object similar to that in `survival`, with the response on the left of the ~ operator being a survival object as returned by the `Surv` function, and the terms on the right being predictors. The arguments `method.beta0`, `beta0`, and `theta0` pertains to the initial starting values. By default, the maximum partial likelihood estimator with the option `MPLE` is used. Otherwise, one can use the ridge estimator with option `ridge`. The `theta0` corresponds to the tuning parameter in ridge estimation. User defined starting values can also be used such as $\boldsymbol{\beta} = \boldsymbol{\gamma} = \mathbf{0}$ by specifying `beta0`. By default, the approximated BIC (Vollinsk and Raftery, 2000) is recommended. However, one can use `AIC` (Akaike, 1974). Alternatively, user-specified penalty is allowed by specifying `lambda0`. The default value for a is n_0. The option `maxit.global` allows for specification of the maximal iteration steps in `SANN` while `maxit.local` specifies the maximal iteration steps for `BFGS`. MIC computes the standard errors (SE) for both $\widetilde{\boldsymbol{\beta}}$ and $\widetilde{\boldsymbol{\gamma}}$. For $\widetilde{\boldsymbol{\beta}}$, the SE computation is only applicable for its nonzero components. The option `maxit.global` asks whether the user wants to output the confidence intervals for γ_j at the confidence level specified by `conf.level` (with 95% as default).

The output of Function `coxphMIC()` is an object of S3 class `coxphMIC`, which is essentially a list of detailed objects that can be used for other purposes. In particular, the item `result` presents the most important results, where one can see the selected model and inference based on testing γ. Two generic functions, `print` and `plot`, are made available for exploring a `coxphMIC` object.

Other R packages for variable selection in Cox PH models

Several other R packages are available for variable selection of Cox PH models. The best subset selection (BSS) is available in the R Package **glmulti** (Calcagno and de Mazancourt, 2010) with AIC only, but it is very slow owing to the intensive computation involved. For large p, a stepwise selection procedure could be used as a surrogate. LASSO (Tibshirani, 1997) can be computed via R Package **glmnet** (Friedman et al., 2010). Zhang and Lu (2007) have made their R codes for implementing ALASSO for Cox models available at `http://www4.stat.ncsu.edu/~hzhang/paper/cox_new.tar`. But the program was written without resorting well to available R routines and it takes an unnecessarily long running time. One alternative way to compute ALASSO is first transform the design matrix $\mathbf{Z} := \mathbf{Z}\,\mathrm{diag}(|\widehat{\boldsymbol{\beta}}|)$ so that LASSO could be applied and then transform the resultant estimates back. SCAD for Cox PH models (Fan and Li, 2002; Fan et al., 2010) can be computed with an earlier version of the R package **SIS** (Saldana and Feng, 2016), but it is no longer available in its current version. One is referred to Table 1, which is presented as Table B1 in Su et al. (2016), for a comparison study of these above-mentioned methods. MIC clearly stands out as the top or among-the-top performer in both sparse estimation and computing time.

Examples

We illustrate the usage of coxphMIC via an analysis of the PBC (primary biliary cirrhosis) data, available from the **survival** package (Therneau and Grambsch, 2000).

Data preparation

To proceed, some minor data preparation is needed. First of all, we want to make sure that the censoring indicator is 0-1 binary.

```
> library(survival); data(pbc);
> dat <- pbc; dim(dat);
[1] 418  20
> dat$status <- ifelse(pbc$status == 2, 1, 0)
```

Next, we explicitly created dummy variable for categorical variables. The `factor()` function could be used instead. Also, grouped sparsity could be used to handle these dummy variables so that they are either all selected or all excluded. We plan to explore this possibility in future research.

```
> dat$sex <- ifelse(pbc$sex == "f", 1, 0)
```

Another necessary step is to handle missing values. This current version does not automatically treat missings. Here, the listwise deletion is used so that only the 276 subjects with complete records are used for further analysis.

```
> dat <- na.omit(dat);
> dim(dat);
[1] 276  20
> head(dat)
  id time status trt      age sex ascites hepato spiders edema bili chol
1  1  400      1   1 58.76523   1       1      1       1   1.0 14.5  261
2  2 4500      0   1 56.44627   1       0      1       1   0.0  1.1  302
3  3 1012      1   1 70.07255   0       0      0       0   0.5  1.4  176
4  4 1925      1   1 54.74059   1       0      1       1   0.5  1.8  244
5  5 1504      0   2 38.10541   1       0      1       1   0.0  3.4  279
7  7 1832      0   2 55.53457   1       0      1       0   0.0  1.0  322
  albumin copper alk.phos    ast trig platelet protime stage
1    2.60    156   1718.0 137.95  172      190    12.2     4
2    4.14     54   7394.8 113.52   88      221    10.6     3
3    3.48    210    516.0  96.10   55      151    12.0     4
4    2.54     64   6121.8  60.63   92      183    10.3     4
5    3.53    143    671.0 113.15   72      136    10.9     3
7    4.09     52    824.0  60.45  213      204     9.7     3
```

The data set now contains 20 variables. Except `id`, `time`, and `status`, there are a total of 17 predictors.

MIC starting with MPLE

To apply coxphMIC, one simply proceeds in the usual way of using coxph formula. By default, all predictors are standardized; the approximated BIC ($\lambda_0 = \ln(n_0)$ is used with $a = n_0$; and the MPLE is used as the starting point.

```
> fit.mic <- coxphMIC(formula = Surv(time, status)~.-id, data = dat, CI.gamma = FALSE)
> names(fit.mic)
 [1] "opt.global" "opt.local"  "min.Q"      "gamma"      "beta"       "VCOV.gamma"
 [7] "se.gamma"   "se.beta"    "BIC"        "result"     "call"
```

The output of coxphMIC contains the minimized Q_n value, the final estimates of γ and $\boldsymbol{\beta}$, the variance-covariance matrix and SE for $\widetilde{\gamma}$, SE for nonzero $\widetilde{\boldsymbol{\beta}}$, BIC value for the final model, and a summary table `result`. In order for the user to be able to inspect the convergence and other detailed info of the optimization algorithms, we also output two objects `opt.global` and `opt.local`, which result from the global (SANN by default) and local optimization (BFGS by default) algorithms.

The output `fit.mic` is a S3 object of coxphMIC class. Two generic functions, `print` and `plot`, are available. The `print` function provides a summary table as below:

```
> print(fit.mic)
           beta0   gamma se.gamma  z.stat p.value beta.MIC se.beta.MIC
trt      -0.0622  0.0000   0.1071  0.0000  1.0000   0.0000          NA
age       0.3041  0.3309   0.1219  2.7138  0.0067   0.3309      0.1074
sex      -0.1204  0.0000   0.1086 -0.0002  0.9998   0.0000          NA
ascites   0.0224  0.0000   0.0991  0.0000  1.0000   0.0000          NA
hepato    0.0128  0.0000   0.1259  0.0000  1.0000   0.0000          NA
spiders   0.0460  0.0000   0.1118 -0.0001  1.0000   0.0000          NA
edema     0.2733  0.2224   0.1066  2.0861  0.0370   0.2224      0.0939
bili      0.3681  0.3909   0.1142  3.4237  0.0006   0.3909      0.0890
chol      0.1155  0.0000   0.1181  0.0002  0.9999   0.0000          NA
albumin  -0.2999 -0.2901   0.1248 -2.3239  0.0201  -0.2901      0.1103
copper    0.2198  0.2518   0.1050  2.3986  0.0165   0.2518      0.0868
alk.phos  0.0022  0.0000   0.0837  0.0000  1.0000   0.0000          NA
ast       0.2308  0.2484   0.1128  2.2023  0.0276   0.2484      0.1025
trig     -0.0637  0.0000   0.0858  0.0000  1.0000   0.0000          NA
platelet  0.0840  0.0000   0.1129  0.0000  1.0000   0.0000          NA
protime   0.2344  0.2293   0.1046  2.1917  0.0284   0.2293      0.1022
stage     0.3881  0.3692   0.1476  2.5007  0.0124   0.3692      0.1243
```

The above results are presented as Table 4 in Su et al. (2016). In this example, MIC started with MPLE given by the first column named `beta0`. Columns 2–5 present estimation of γ and the hypothesis testing results on $H_0 : \gamma_j = 0$. The estimates of $\boldsymbol{\beta}$ are given in the last two columns. It can be seen that eight variables are selected in the final model, which are `age`, `edema`, `bili`, `albumin`, `copper`, `ast`, `protime`, and `stage`.

The `plot` function provides error bar plots based on the MIC estimator of both $\boldsymbol{\beta}$ and the reparameterized γ :

```
> plot(fit.mic, conf.level = 0.95)
```

as shown in Figure 2. Essentially, the 95% confidence intervals (CI) are plotted. One can modify the confidence level with the `conf.level` option. To compare two plots conveniently, they are made with the same range on the vertical y-axis. Note that CI is not available for any zero β_j estimate in Panel (b), which corresponds to an unselected variable. Those selected variables are highlighted in green color in Panel.

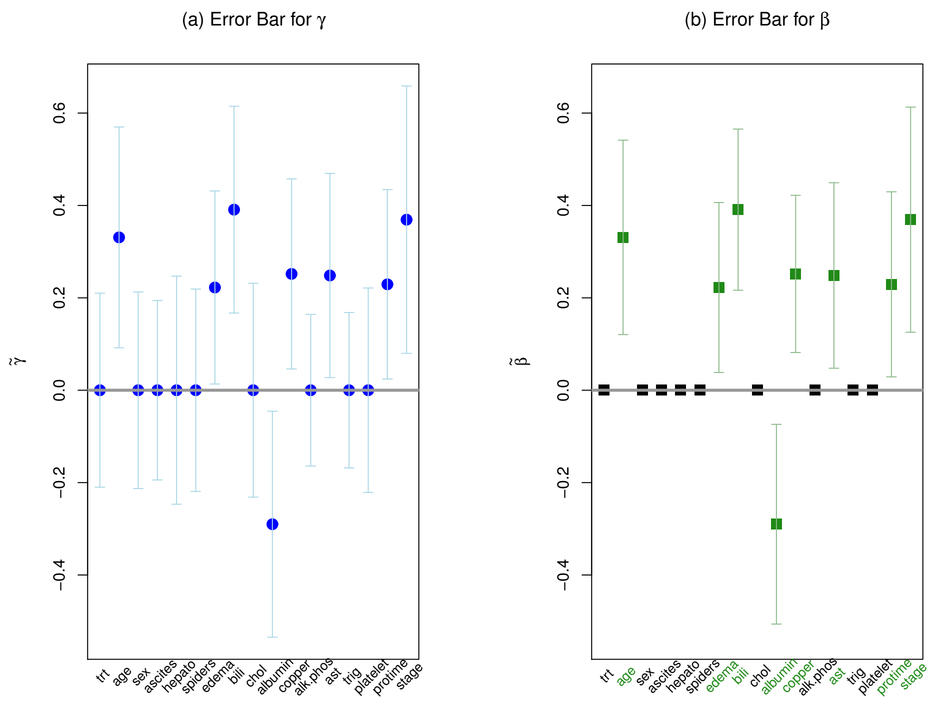

Figure 2: Error bar plots for MIC estimates of γ in (a) and β in (b). The 95% confidence intervals (CI) are plotted. The selected variables are highlighted in green in Panel (b).

Multiple starting points

Trying out multiple starting point is a common strategy in facing global optimization problems. We may consider starting with the **0** vector, which corresponds to the null model. Having `beta0 = 0` is actually the default option if `method.beta0` is neither '`MPLE`' nor '`ridge`' and a specific value for `beta0` is not given, i.e., setting `beta0 = NULL`.

```
> fit0.mic <- coxphMIC(formula = Surv(time, status)~.-id, data = dat,
+ method = "BIC", scale.x = TRUE, method.beta0 = "zero")

> c(fit.mic$min.Q, fit0.mic$min.Q)
[1] 974.3340 978.1232
```

We can compare the minimized objective function `min.Q` to decide which fitting result is preferable (i.e., the smaller one). The above result suggests that the fit with MPLE as starting point remains preferable.

Concerning sparse estimation, the vectors with 0/+1/-1 values obtained by applying a threshold to the MPLE $|\widehat{\boldsymbol{\beta}}|$ could be reasonable choices for the starting point too, i.e.,

$$\beta_{0j} := \text{sgn}(\hat{\beta}_j)\, I\left\{|\hat{\beta}_j| > c_0\right\},$$

where $c_0 > 0$ is a threshold close to 0. For example, setting $c = 0.06$ yields

```
> beta.MPLE <-  fit.mic$result[, 1]
> beta0 <- sign(beta.MPLE)*sign(abs(beta.MPLE) > .06);
> cbind(beta.MPLE, beta0)
      beta.MPLE beta0
 [1,]   -0.0622    -1
 [2,]    0.3041     1
 [3,]   -0.1204    -1
 [4,]    0.0224     0
 [5,]    0.0128     0
 [6,]    0.0460     0
 [7,]    0.2733     1
 [8,]    0.3681     1
 [9,]    0.1155     1
[10,]   -0.2999    -1
[11,]    0.2198     1
[12,]    0.0022     0
[13,]    0.2308     1
[14,]   -0.0637    -1
[15,]    0.0840     1
[16,]    0.2344     1
[17,]    0.3881     1
```

In the above example, we applied a threshold of 0.06 to the MPLE to obtain a 0/+1/-1 valued vector. To start with this user-supplied starting point, one proceeds as follows.

```
> fit1.mic <- coxphMIC(formula = Surv(time, status)~.-id, data = dat,
+ method = "BIC", scale.x = TRUE, method.beta0 = "user-supplied", beta0 = beta0)
> c(fit.mic$min.Q, fit0.mic$min.Q, fit1.mic$min.Q)
[1] 974.3340 978.1232 979.6826
```

Again, the fitting starting at MPLE seems the best in this example, by giving the smallest minimized value.

Different a values

We may consider obtaining the regularization path with respect to a. According to asymptotic results, $a = O(n)$ is desirable and the recommended value is $a = n_0$ the number of uncensored deaths, which is $n_0 = 111$ in the PBC data under study.

We try out a spread of a values that range from 10 to 200, as prescribed by the R object `A0`.

```
> set.seed(818)
> n <- NROW(dat); n0 <- sum(dat$status == 1)
> A0 <- 10:200
> p <- NCOL(dat)-3
> BETA <- matrix(0, nrow = length(A0), ncol = p)   # USE ARRAY
> for (j in 1:length(A0)){
    su.fit <- coxphMIC(formula = Surv(time, status)~.-id, data = dat, a0 = A0[j],
        method = "BIC", scale.x = TRUE)
    BETA[j, ] <- su.fit$beta
  }
> BETA <- as.data.frame(BETA)
> colnames(BETA) <- colnames(dat)[-(1:3)]
> row.names(BETA) <- A0
> head(BETA, n = 5)
   trt    age sex ascites hepato spiders  edema   bili chol albumin copper alk.phos
10   0 0.2983   0       0      0       0 0.2024 0.4135    0 -0.2799 0.2495        0
11   0 0.2987   0       0      0       0 0.2015 0.4159    0 -0.2799 0.2491        0
12   0 0.2992   0       0      0       0 0.2006 0.4181    0 -0.2799 0.2487        0
13   0 0.3000   0       0      0       0 0.1998 0.4200    0 -0.2801 0.2482        0
14   0 0.3009   0       0      0       0 0.1992 0.4216    0 -0.2804 0.2478        0

      ast trig platelet protime  stage
10 0.1937    0        0  0.1912 0.3583
11 0.1924    0        0  0.1895 0.3612
12 0.1914    0        0  0.1878 0.3642
13 0.1906    0        0  0.1862 0.3672
14 0.1901    0        0  0.1847 0.3701
```

A plot of the regularization path with respect to *a*, as shown in Figure 3, can be obtained as follows:

```
> par(mar = rep(5,4), mfrow = c(1,1))
> x.min <- min(A0); x.max <- max(A0)
> plot(x = c(x.min, x.max), y = c(min(BETA), max(BETA)), type = "n",
+    xlab = "a", cex.lab = 1.2, las = 1, ylab = expression(tilde(beta)))
> for (j in 1:ncol(BETA)){
+    lines(x = A0, y = BETA[,j], col = "red", lty = 1, lwd = 1)
+    points(x = A0, y = BETA[,j], col = "red", pch = j, cex = .3)
+    vname <- colnames(BETA)[j]
+    if (abs(BETA[nrow(BETA),j]) > .00001) {
        # text(x.max+5, BETA[nrow(BETA),j], labels = vname, cex = 1, col = "blue")
+        mtext(text = vname, side = 4, line = 0.5, at = BETA[nrow(BETA),j], las = 1,
+            cex = 1, col = "blue", font = 1)
+    }
+ }
> abline(h = 0, col = "gray25", lwd = 2)
> abline(v = n0, col = "gray45", lwd = 1.5)
> text(n0+5, -0.2, expression(paste("a = ", n[0], " = ", 111, sep = "")), cex = 1.2,
+ col = "gray35")
```

From Figure 3, it can be seen that the regularization path is essentially flat with respect to *a*, especially for relatively large *a* values. This indicates that treating *a* as a tuning parameter is unnecessary.

Summary

The paper presents the `coxphMIC` package to implement the MIC method for Cox proportional hazards models. Compared to several other competitive methods, MIC has three main advantages by offering a superior empirical performance for it aims to minimize BIC (albeit approximated) without reducing the search space, great computational efficiency since it does not involve selection of any tuning parameter, and a leeway to perform significance testing that is free of the post-selection inference.

Bibliography

H. Akaike. A new look at model identification. *IEEE Transactions an Automatic Control*, 19(6):716–723, 1974. [p1, 4]

V. Calcagno and C. de Mazancourt. glmulti: An R package for easy automated model selection with (generalized) linear models. *Journal of Statistical Software*, 34(12), 2010. [p4]

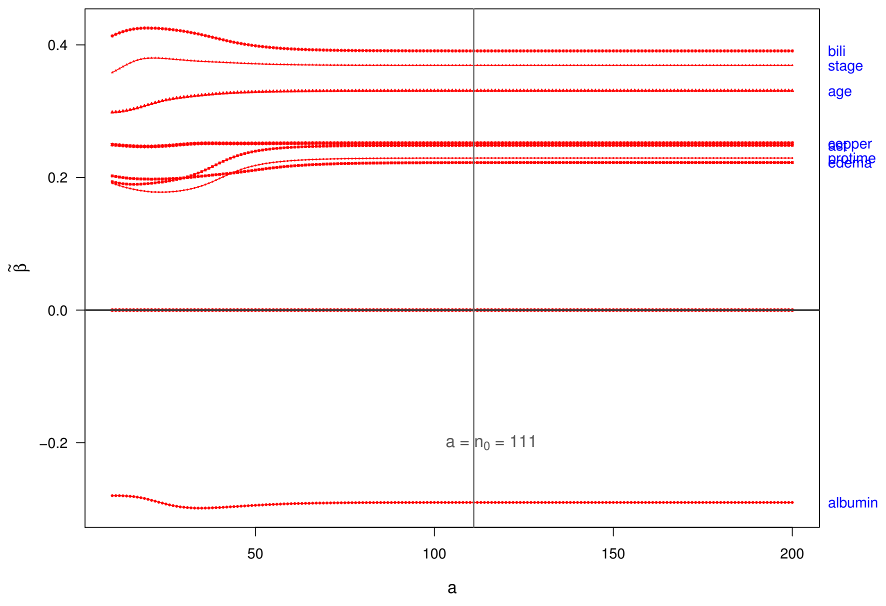

Figure 3: The regularization path with respect to a with the PBC data. The a value varies from 10 to 200. The recommended value is $a = n_0 = 111$.

D. R. Cox. Regression models and life tables (with discussion). *Journal of the Royal Statistical Society, Series B*, 34(2):187–220, 1972. [p1]

D. R. Cox. Partial likelihood. *Biometrika*, 62(2):269–276, 1975. [p1]

J. Fan and R. Li. Variable selection for cox's proportional hazards model and frailty model. *Annals of Statistics*, 30(1):74–99, 2002. [p3, 4]

J. Fan, Y. Feng, and Y. Wu. High-dimensional variable selection for cox's proportional hazards model. *The Institute of Mathematical Statistics*, 6:70–86, 2010. [p4]

J. H. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1), 2010. [p4]

H. Leeb and B. M. Pötscher. Model selection and inference: facts and fiction. *Econometric Theory*, 21(1): 21–59, 2005. [p1]

K. Makiyama, A. Hayakawa, S. Uryu, and H. Yutani. `githubinstall`: An easy way to install R packages on github. 2016. URL `https://cran.r-project.org/web/packages/githubinstall`. URL `https://cran.r-project.org/web/packages/githubinstall/`. [p]

K. M. Mullen. Continuous global optimization in R. *Journal of Statistical Software*, 60(6), 2014. URL `https://www.jstatsoft.org/article/view/v060i06/v60i06.pdf`. [p3]

D. F. Saldana and Y. Feng. SIS: An R package for sure independence screening in ultrahigh-dimensional statistical models. *Journal of Statistical Software*, 2016. URL `http://www.stat.columbia.edu/~diego/PapersandDraft/SIS.pdf`. In press. [p4]

G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978. [p1]

X. Su. Variable selection via subtle uprooting. *Journal of Computational and Graphical Statistics*, 24(4): 1092–1113, 2015. [p2, 3]

X. Su, C. S. Wijayasinghe, J. Fan, and Y. Zhang. Sparse estimation of cox proportional hazards models via approximated information criteria. *Biometrics*, 72:751–759, 2016. [p1, 2, 4, 6]

T. Therneau and P. Grambsch. Modeling survival data: Extending the cox model. *Springer-Verlag*, 2000. [p3, 5]

R. Tibshirani. The LASSO method for variable selection in the Cox model. *Statistics in Medicine*, 16(4): 385–395, 1997. [p4]

C. T. Vollinsk and A. E. Raftery. Bayesian information criterion for censored survival models. *Biometrics*, 56(1):256–262, 2000. [p1, 3, 4]

H. Zhang and W. Lu. Adaptive lasso for Cox's proportional hazards model. *Biometrika*, 94(3):691–703, 2007. [p4]

Razieh Nabi
Department of Computer Science
Johns Hopkins University
Maryland 21218, USA
rnabiab1@jhu.edu

Xiaogang Su
Department of Mathematical Sciences
University of Texas at El Paso
Texas 79968, USA
xsu@utep.edu