Computer Methods and Programs in Biomedicine 168 (2019) 21–37

Contents lists available at ScienceDirect

Computer Methods and Programs in Biomedicine

journal homepage: www.elsevier.com/locate/cmpb

compound.Cox: Univariate feature selection and compound covariate for predicting survival

Takeshi Emura[a,*], Shigeyuki Matsui[b], Hsuan-Yu Chen[c]

[a] Graduate Institute of Statistics, National Central University, Zhongda Road, Zhongli District, Taoyuan 32001, Taiwan
[b] Department of Biostatistics, Nagoya University Graduate School of Medicine, 65 Tsurumai-cho, Showa-ku, Nagoya, 466-8550, Japan
[c] Institute of Statistical Science, Academia Sinica, 128 Academia Road Sec.2, Nankang Taipei 115, Taiwan

ARTICLE INFO

Article history:
Received 28 March 2018
Revised 26 September 2018
Accepted 26 October 2018

Keywords:
Cancer prognosis
Copula
Cox regression
Cross-validation
Dependent censoring
False discovery rate
Gene expression
High-dimensional data
Multiple testing

ABSTRACT

Background and objective: Univariate feature selection is one of the simplest and most commonly used techniques to develop a multigene predictor for survival. Presently, there is no software tailored to perform univariate feature selection and predictor construction.

Methods: We develop the *compound.Cox* R package that implements univariate significance tests (via the Wald tests or score tests) for feature selection. We provide a cross-validation algorithm to measure predictive capability of selected genes and a permutation algorithm to assess the false discovery rate. We also provide three algorithms for constructing a multigene predictor (compound covariate, compound shrinkage, and copula-based methods), which are tailored to the subset of genes obtained from univariate feature selection. We demonstrate our package using survival data on the lung cancer patients. We examine the predictive capability of the developed algorithms by the lung cancer data and simulated data.

Results: The developed R package, *compound.Cox*, is available on the CRAN repository. The statistical tools in *compound.Cox* allow researchers to determine an optimal significance level of the tests, thus providing researchers an optimal subset of genes for prediction. The package also allows researchers to compute the false discovery rate and various prediction algorithms.

1. Introduction

Univariate feature selection is one of the simplest and most commonly used techniques to develop a multigene predictor for survival of cancer patients. It picks up genes having P-values lower than a cutoff in testing association between genes and survival under univariate Cox models [1]. A predictor constructed from the selected genes is useful for predicting survival in various cancers [2-11]. To construct a predictor, it is essential to determine the number of genes to be included in the predictor. Researchers often set a fixed threshold (e.g., P-value < 0.001), or a data-driven threshold that optimizes predictive capability.

By adopting the univariate tests, one would select features individually associated with survival. More elaborate multivariate feature selection methods accounting for correlations between features can also be considered, but their advantages over the simpler univariate methods have not convincingly been demonstrated through many survival prediction analyses of cancer datasets [1,12,13,14]. Methods ignoring correlations between features, such as diagonal linear discriminant analysis [15] and compound covariate [11,16], performs reasonably well in some simulation studies. In addition, the univariate tests adapt well to clinical trials involving the development of genomic signatures [10,17].

While univariate feature selection is a simple approach, a variety of computing algorithms should be implemented:

(i) Computation of Z-values and P-values (via the Wald tests or score tests),
(ii) Choice of a P-value threshold,
(iii) Assessment of predictive capability of selected genes,
(iv) Assessment of false discoveries (falsely selected genes),
(v) Construction of a multigene predictor after selection.

As for choosing a P-value threshold, the simplest approach is to apply the traditional value of 0.05, 0.01, or 0.001 [3,4,7-10]. A more refined approach for choosing a threshold is to optimize a cross-validated predictive criterion [6,18,19]. The computational algorithms also vary in terms of predictor construction after se-

* Corresponding author.
E-mail address: takeshiemura@gmail.com (T. Emura).

https://doi.org/10.1016/j.cmpb.2018.10.020
0169-2607/

lection. These includes the predictor based on a multivariate Cox model [18], the compound covariate predictor [6,10,16], the ridge regression [9,18], and the predictor fitted by a copula model under dependent censoring [20,21].

In this article, we introduce the *compound.Cox* R package [22] that implements univariate significance tests (via the Wald tests or score tests) for feature selection as well as a cross-validation analysis for measuring predictive capability of selected genes. These analytical tools allow users to determine an optimal significance level of the tests, thus providing users an optimal subset of genes for prediction. The tool for evaluating the *false discovery rate* (FDR) is also given in the package. The package provides three methods for constructing a multigene predictor (compound covariate, compound shrinkage, and copula-based methods), which are tailored to the subset of genes obtained from univariate feature selection. To provide a running example, we included survival data on lung cancer patients in the package. We demonstrate the predictive capability of the developed algorithms by the lung cancer data and simulated data.

2. Univariate feature selection

This section reviews the basic background for univariate feature selection methods that have been studied for prognostic prediction of survival.

Univariate feature selection is the traditional method for selecting a subset of genes that are predictive of survival. In the initial step, one fits a univariate Cox model for each gene, one-by-one. Then, one selects a subset of genes that are univariately associated with survival. The subset typically consists of genes that have P-values lower than a threshold. The number of selected genes, denoted by q, is typically $q \leq 100$ and rarely $q > 200$.

2.1. P-value threshold

A P-value threshold can be chosen by a variety of rules. Many medical researchers tend to use the traditional P-value threshold of 0.05 [7] and 0.01 [3,8,9]. Simon [23] recommended the P-value threshold of 0.001 that is often useful in survival analysis [4,10,11]. While the fixed threshold approaches are easy to use, the threshold values are often arbitrary and sub-optimal.

A threshold can be determined by optimizing predictive capability of the selected genes. For the lung cancer data, Beer et al. [2] chose $q=50$ genes that maximized the association between overall survival and the multigene predictor. To predict metastasis of breast cancer patients, Wang et al. [5] chose $q=76$ genes that maximized the area under the ROC curve. These exploratory techniques typically need to examine different thresholds, different tests, and different predictors before reaching a final set of genes.

Matsui [6] recommended a more automatic routine by optimizing a *cross-validated likelihood* (CVL). This method was applied to construct a compound covariate (a multigene predictor) consisting of $q=75$ or 85 genes. A different cross-validated likelihood with a multivariate Cox model was considered in Bøvelstad et al. [18]. However, their optimized set of genes contained only a few genes and the resultant multigene predictor had poor predictive capability. This is because their predictive capability of the optimized set of genes was not properly measured under a multivariate Cox model.

In summary, it is critical to develop a set of algorithms that are tailored for univariate feature selection. Researchers currently should perform a "for loop" algorithm to run the sequence of univariate Cox regressions across all genes. However, this algorithm is not convenient when researchers try many different tests (the Wald tests or score tests) and different P-value thresholds. The computation of the CVL [6] raises another challenge due to its complex algorithm.

2.2. Cross-validation

Simon [24] suggested using cross-validation to estimate a predictive capability of multigene predictors. In a K-fold cross-validation, samples of size n are randomly divided into K groups of approximately equal sizes, which can be indexed by $k=1, \ldots, K$. A feature selection algorithm is applied to the samples without the k-th group ($n-n/K$ training samples). Using the set of selected features, multigene predictors are constructed for the samples in the k-th group (n/K test samples). Then, a measure of predictive accuracy, denoted by CV_k, is computed. Repeat this process for $k=1, \ldots, K$, and compute the overall measure of predictive capability $\sum_{k=1}^{K} CV_k$.

The choice $K=n$ yields the leave-one-out cross-validation that gives the most unbiased estimate of predictive capability. In practice, the number $K=5$ (or $K=10$) is often chosen to reduce computation time. In this case, a predictive capability measure computed by cross-validation varies according to how to divide the samples. Hence, some quality control method is encouraged to assess the random variation.

If the sample size n is small, a larger number for K (e.g., $K=20$) is recommended to ensure that each group has sufficient numbers of samples.

The interpretation of predictive capability with cross-validation remains a challenge when the samples involve high-dimensional features. For each fold of cross-validation, the process of feature selection is implemented from scratch. Consequently, each fold yields its own selected features that are often remarkably different from the selected features by the whole samples. This implies that cross-validation evaluates the "selection algorithm" rather than the "selected features".

2.3. Multigene predictors

A multigene predictor is a weighted sum of gene expressions. A large (small) predictor value corresponds to poor (good) survival, which can be used to classify a patient into a good or poor prognosis group.

To calculate the weights of a multigene predictor, one can use the regression coefficients or Z-values obtained from univariate tests. The resultant predictor is called the *compound covariate* (CC) as proposed by Tukey [25]. The CC predictor is an ensemble of univariate analyses, which does not employ a multivariate model. The CC predictor has been shown to be useful in predicting survival with gene expressions [2,5-7,10,11,13,16].

After feature selection, medical researchers tend to re-fit a multivariate Cox regression model to develop a refined multigene predictor. This is because some genes identified in univariate significance analyses are no longer significant in multivariate analysis [8,26] due to their multicollinearity and high-dimensionality. However, if non-significant features in a multivariate Cox model are simply removed, the resultant predictor has poor predictive capability [18,19]. Ridge regression is an approach to combat both multicollinearity and high-dimensionality [18]. For instance, to predict survival of ovarian cancer patients, Yoshihara et al. [9] developed a ridge-based predictor after selecting $q=88$ genes by univariate feature selection. The compound shrinkage (CS) estimator [16] provides another strategy to refine a multigene predictor, which adjusts the CC predictor by incorporating multivariate likelihood information.

2.4. Dependent censoring

The standard univariate feature selection methods may produce biased results if censoring is due to informative dropout [21,27]. This is because Cox regression requires the independent censoring assumption. The issues of dependent censoring have been intensively discussed in the literature, and now several statistical methods can remedy the issues [20,21,27-32]. Nonetheless, these methods have not been widely used due to the requirement of mathematical skills and high computational time. Emura and Chen [20] proposed a copula-based method to perform univariate feature selection, where a copula can adjust for the biased results due to dependent censoring. We shall introduce this method as a component of *compound.Cox*.

3. The compound.Cox package

The basis of *compound.Cox* is the sequence of univariate Cox models

$$h_j(t|x_j) = h_{0j}(t)\exp(\beta_j x_j),\ j = 1,\ \ldots,\ p,$$

where x_j is the j-th feature (gene), $h_{0j}(\cdot)$ is the baseline hazard function, and p is the number of features. The *compound.Cox* package performs feature selection through the multiple tests for H_{0j}: $\beta_j = 0$ vs. H_{1j}: $\beta_j \neq 0$ for $j = 1,\ldots,\ p$, where p can be large (e.g., $p = 100$ and $p = 5000$). Features are selected according to the significance level for the test results.

We first introduce the lung cancer data made available in *compound.Cox*.

3.1. The lung cancer data

Chen et al. [7] analyzed the data from 125 lung cancer patients whose gene expressions were coded as 1, 2, 3, or 4 (~25th, 25th~50th, 50th~75th, or 75th~ percentile). These values were treated as continuous covariates (not as factors). The primary endpoint is overall survival (i.e., time-to-death). During the follow-up, 38 patients died and the remaining 87 patients were censored. In Chen et al. [7], the 125 patients were separated into 63 training and 62 testing samples. Univariate feature selection performed on the 63 training samples resulted in 16 genes that are predictive of survival (P-value < 0.05 in the Wald tests).

We made the subset of the lung cancer data available in *compound.Cox*. The subset contains $p = 97$ gene expressions which are associated with overall survival (P-value < 0.20 in the Wald tests). The data are stored in the *Lung* object, a data-frame with 125 samples (patients):

In the outputs above, the variables are defied as

- t.vec: survival time (time to either death or censoring) in months
- d.vec: censoring indicator; 1 = death, or 0 = censoring
- train: index for training sample; TRUE = training sample, or FALSE = testing sample
- VHL: gene expression, coded as 1, 2, 3, or 4
- IHPK1: gene expression, coded as 1, 2, 3, or 4
- RPL5: gene expression, coded as 1, 2, 3, or 4

Whilst $p = 97$ is not as high-dimensional as those commonly seen in microarray analyses, the data allow users to run our illustrative R codes in a reasonable amount of computing time. Note that gene expressions are usually continuously valued in contrast to the ordinal coding values of 1, 2, 3, or 4 in the above example.

Supplementary Material includes the R codes to analyze the lung cancer data using the *compound.Cox* package.

3.2. Data input

The form of survival data in *compound.Cox* is $\{(t_i, \delta_i, \mathbf{x}_i);\ i = 1, \ldots,\ n\}$, where

- t_i: survival time or censoring time,
- δ_i: censoring indicator ($\delta_i = 1$ if t_i is survival time, or $\delta_i = 0$ if t_i is censoring time),
- $\mathbf{x}_i = (x_{i1}, \ldots,\ x_{ip})$: p-dimensional features (genes),

and n is the number of samples. This is the standard form of survival data except that p is allowed to be greater than n. In *compound.Cox*, we use the following styles for inputs:

- t.vec: a vector $(t_1, t_2, \ldots,\ t_n)$,
- d.vec: a vector $(\delta_1, \delta_2, \ldots,\ \delta_n)$,
- X.mat: a matrix with the i-th row $\mathbf{x}_i = (x_{i1}, \ldots,\ x_{ip})$ for $i = 1,\ \ldots,\ n$.

One can create t.vec, d.vec, and X.mat for the 63 training samples by using the R codes in Supplementary Material.

3.3. Feature selection

The function *uni.selection(, P.value =, score =)* can perform univariate feature selection by setting a threshold "*P.value = P*" for $0 < P < 1$. For instance, $P = 0.05$ is the traditional 5% significance level. It incorporates the Wald test (*score = FALSE*) and score test (*score = TRUE*) into a single function.

```
> Lung
     t.vec       d.vec  train  VHL  IHPK1  HMMR  CMKOR1  PLAU  IGF2  FGB  ..... RPL5
1    47.0627063  0      FALSE  2    2      2     3       4     1     1          4
2    49.2739274  0      TRUE   3    4      1     3       4     3     1          4
3    20.0660066  1      TRUE   2    3      1     3       4     4     3          1
4    26.9966997  1      TRUE   2    4      1     4       4     4     1          2
5    39.9009901  0      FALSE  3    4      2     3       3     1     1          4
6    35.9380152  0      TRUE   3    3      2     3       4     1     1          4
7    45.2805281  0      TRUE   4    2      1     2       3     4     1          4
8    9.4389439   1      TRUE   4    2      3     4       1     1     3          4
9    21.2871287  1      FALSE  4    3      3     3       4     3     2          4
10   41.3201320  0      TRUE   4    4      4     4       4     4     4          4
.
.
.
125  56.841411   0      FALSE  3    2      2     2       4     1     1          3
```

- Wald tests

Define a univariate estimator $\hat{\beta}_j = \arg\max \ell_j(\beta_j)$, where $\ell_j(\cdot)$ is the log likelihood defined as

$$\ell_j(\beta_j) = \sum_{i=1}^{n} \delta_i \left[\beta_j x_{ij} - \log\left(\sum_{\ell \in R_i} \exp(\beta_j x_{\ell j}) \right) \right], \quad R_i = \{ \ell;\ t_\ell \geq t_i \}.$$

where $\boldsymbol{\delta} = (\delta_1, \ldots, \delta_n)'$ is a vector of censoring indicators, $\mathbf{X}$ is a $n \times p$ matrix with elements x_{ij}, and $\mathbf{S}^{(k)}$is a $n \times p$ matrix with elements $S_{ij}^{(k)}$. Thus, the vector of Z-statistics is $\mathbf{S}/\mathbf{V}^{1/2}$. Note that this vector-based computing scheme is efficiently programmed in R.

The function *uni.selection()* performs feature selection for a given value of *P*. Below, one can see the results of feature selection using the Wald tests with $P = 0.05$. In the input below, "score = FALSE" means that the Wald tests are used instead of the score tests.

```
> uni.selection(t.vec,d.vec,X.mat,K=20,P.value=0.05,score=FALSE,permutation=TRUE) ## Wald test
$beta
   ANXA5     DLG2    ZNF264    DUSP6    CPEB4      LCK     STAT1
-1.0876762 1.3215044 0.5473276 0.7524497 0.5891676 -0.8447389 -0.5844262
    RNF4     IRF4     STAT2      HGF     ERBB3      NF1     FRAP1
 0.6463635 0.5176704 0.5849869 0.5086750 0.5509026 0.4715235 -0.7696768
    MMD      HMMR
 0.9151541 0.5156711

$Z
   ANXA5     DLG2    ZNF264    DUSP6    CPEB4      LCK     STAT1     RNF4
-2.885540 2.872880 2.654412 2.628478 2.404015 -2.384028 -2.329287 2.290596
    IRF4     STAT2      HGF     ERBB3      NF1     FRAP1      MMD     HMMR
2.171948 2.155568 2.127643 2.126139 2.074913 -2.045298 2.034407 1.976606

$P
     ANXA5       DLG2       ZNF264       DUSP6       CPEB4         LCK
0.003907424 0.004067486 0.007944666 0.008576790 0.016216117 0.017124302
     STAT1       RNF4       IRF4       STAT2       HGF         ERBB3
0.019843870 0.021986777 0.029859561 0.031117422 0.033366690 0.033491656
     NF1       FRAP1       MMD       HMMR
0.037994593 0.040825466 0.041910555 0.048086199

$CVL
     CVL      RCVL1      RCVL2
-98.66365 -86.88157 -87.75034

$Genes
No. of genes  No. of selected genes  No. of falsely selected genes
      97.000                16.000                         5.005

$FDR
P.value * (No. of genes)  Permutation
               0.3031250    0.3128125
```

The Wald tests are based on the Z-values $z_j = \hat{\beta}_j / SE(\hat{\beta}_j)$, $j = 1, \ldots, p$, where $SE(\beta_j) = [-\partial^2 \ell_j(\beta_j)/\partial \beta_j^2]^{-1/2}$. Under the null hypothesis H_{0j}: $\beta_j = 0$, the distribution of z_j is approximated by the standard normal distribution $Z \sim N(0, 1)$. The selected genes are represented as a set $\Omega = \{ j:\ P_j < P \}$, where $P_j = \Pr(|Z| > |z_j|)$ is the P-value for testing H_{0j}: $\beta_j = 0$ vs. H_{1j}: $\beta_j \neq 0$.

- Score tests

The univariate score statistics and its variances are

$$S_j = \sum_{i=1}^{n} \delta_i \left(x_{ij} - S_{ij}^{(1)}/S_{ij}^{(0)} \right), \qquad V_j = \sum_{i=1}^{n} \delta_i \left(S_{ij}^{(2)}/S_{ij}^{(0)} - (S_{ij}^{(1)}/S_{ij}^{(0)})^2 \right),$$

where $S_{ij}^{(k)} = \sum_{\ell \in R_i} x_{\ell j}^k$ for $k = 0$, 1 or 2, and $j = 1, \ldots, p$. The Z-value is obtained by $z_j = S_j / V_j^{1/2}$. Witten and Tibshirani [1] stabilized the Z-value by using a constant $d_0 > 0$ such that $z_j^{d_0} = S_j / (V_j^{1/2} + d_0)$. The value $d_0 = 0$ reduces to the score statistics. The selected genes are represented as a set $\Omega = \{ j:\ P_j < P \}$, where P_j is the P-value for testing H_{0j}: $\beta_j = 0$ vs. H_{1j}: $\beta_j \neq 0$. The vector of univariate score statistics and the vector of their variances are:

$$\mathbf{S} = \boldsymbol{\delta}' \left(\mathbf{X} - \mathbf{S}^{(1)}/\mathbf{S}^{(0)} \right), \qquad \mathbf{V} = \boldsymbol{\delta}' \left(\mathbf{S}^{(2)}/\mathbf{S}^{(0)} - \left(\mathbf{S}^{(1)}/\mathbf{S}^{(0)} \right)^2 \right),$$

The outputs show $\hat{\beta}_j$, z_j, and P_j for each of the selected features.

The outputs also show the CVL value that is defined in Appendix A. A high CVL value corresponds to a better predictive capability for the selected features. The input "$K = 20$" means that 20-fold cross-validation is applied to compute the CVL value. Since the data have small sample sizes ($n = 63$), the usual choice of $K = 5$ or $K = 10$ leads to unstable results.

The outputs also show the FDR that is defined in Appendix B. A low FDR value corresponds to a lower proportion of falsely selected features. The input "permutation = TRUE" means that the permutation method of Appendix B is used to compute the FDR. One could save computational time by setting "permutation = FALSE" .

The Wald tests selected 16 genes (P-value < 0.05, CVL $= -98.66$, FDR $= 0.30$ or 0.31). The FDR of 0.31 implies that there are $16 \times 0.31 \approx 5$ falsely selected genes out of the 16 selected genes. The score tests selected 18 genes (P-value < 0.05, CVL $= -97.37$, FDR $= 0.27$ or 0.31); the outputs for the score tests are given in Appendix C. Hence, the score tests had slightly better CVL and FDR values than the Wald tests do. The selected genes are sorted according to their P-values.

In the outputs, "CVL = −98.66" is seen together with "RCVL1 = −86.88" and "RCVL2 = −87.75". RCVL1 and RCVL2 are named "re-substitution CVL", which provide upper control limits for the variation of the CVL value due to random cross-validation. If the CVL value is less than the RCVL1 and RCVL2 values, then the variation of the CVL value is in-control. In the outputs, we conclude that "CVL = −98.66" is in-control since CVL < RCVL1 and CVL < RCVL2. Appendix A defines RCVL1 and RCVL2 and explains how to interpret them.

3.5. Significance tests

We have seen that *uni.selection()* performs multiple tasks – selection of features, sorting of selected features, and computation of the CVL and FDR values. However, some users simply wish to perform significance tests for all features without selection or sorting. For this reason, we prepare more elementary functions, *uni.Wald()* and *uni.score()*. These functions also save computing time by avoiding the complex algorithms for the CVL and FDR.

In the outputs below, we use "res" to store the values of $\hat{\beta}_j$, z_j, and P_j for all the features under the Wald tests. Then we extract $\hat{\beta}_j$ for the selected features with P-value < 0.05.

```
res=uni.Wald(t.vec,d.vec,X.mat)
> res$beta[res$P<0.05] ## Wald tests
HMMR       LCK        ANXA5      IRF4       STAT2      ERBB3      NF1
0.5156711  -0.8447389 -1.0876762 0.5176704  0.5849869  0.5509026  0.4715235
DLG2       HGF        CPEB4      ZNF264     MMD        RNF4       FRAP1
1.3215044  0.5086750  0.5891676  0.5473276  0.9151541  0.6463635  -0.7696768
STAT1      DUSP6
-0.5844262 0.7524497
```

3.4. Prediction after selection

Consider a test sample having the q selected features (x_1,..., x_q). To predict survival of this sample, we construct a predictor $w_1x_1 + \cdots + w_qx_q$ whose high (low) value is associated with poor (good) prognosis for survival.

Using the outputs for the Wald tests, we set the weights $w_j = \hat{\beta}_j$ attached to the $q = 16$ selected genes. We then arrive at the 16-gene predictor of Chen et al. [7]:

$$\begin{aligned}\text{The } 16-\text{genepredictor} &= (-1.09 \times \text{ANXA5}) + (1.32 \times \text{DLG2})\\ &+(0.55 \times \text{ZNF264}) + (0.75 \times \text{DUSP6})\\ &+(0.59 \times \text{CPEB4}) + (-0.84 \times \text{LCK}) + (-0.58 \times \text{STAT1})\\ &+(0.65 \times \text{RNF4}) + (0.52 \times \text{IRF4})\\ &+(0.58 \times \text{STAT2}) + (0.51 \times \text{HGF}) + (0.55 \times \text{ERBB3})\\ &+(0.47 \times \text{NF1}) + (-0.77 \times \text{FRAP1})\\ &+(0.92 \times \text{MMD}) + (0.52 \times \text{HMMR}).\end{aligned}$$

The resultant predictor is a compound covariate (CC) since it is an ensemble of univariate analyses, which does not employ a multivariate analysis [6,16].

To construct a predictor based on the score tests, we use the Z-value of the univariate score tests $w_j = z_j$ attached to the $q = 18$ selected genes [6]. Usually, z_j has the same sign as $\hat{\beta}_j$.

The computational speed of *uni.score()* is much faster than that of *uni.Wald()* since the score test does not estimate β_j. In *uni.score()*, we provide the one-step estimator S_j/V_j that gives an approximate value of $\hat{\beta}_j$. Consequently, the outputs for the score tests are similar to those for the Wald tests.

3.6. Optimizing a threshold by CVL

The optimal P-value threshold can be determined by plotting of the CVL values against P-value thresholds. Since the large CVL value corresponds to better predictive capability, one can determine the P-value threshold by searching the maxima of the CVL plot.

Fig. 1 draws the CVL plot for the lung cancer data. It shows that the optimal threshold is "P-value < 0.0205" for the Wald tests and the optimal threshold is "P-value < 0.0275" for the score tests. The CVL plots are not very smooth due to the small sample size ($n = 63$). Fig. 1 also draws the plots for RCVL1 and RCVL2, which provide the upper control limits for the CVL plot. We observe that the CVL plot is in-control since it is under the RCVL1 and RCVL2 plots.

Supplementary Material provides the R codes for creating the CVL plot and its upper control limits (S1 for the lung cancer data; S2 for simulated data).

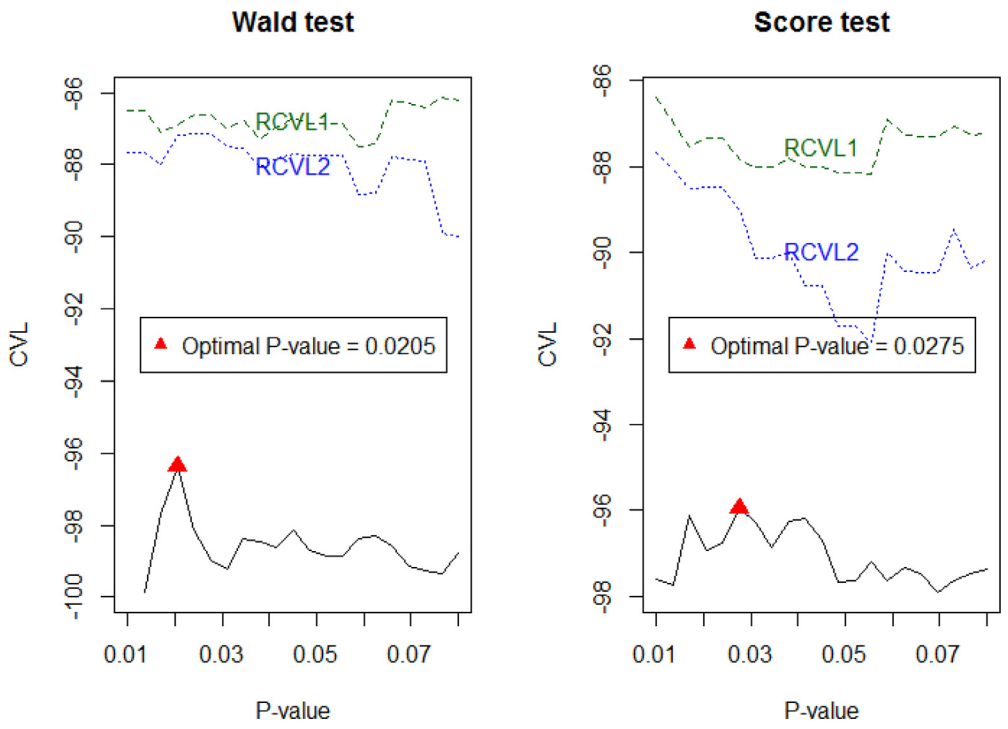


Fig. 1. The plot of the CVL values against P-value thresholds. Larger CVL values correspond to better predictive ability of selected features. RCVL1 and RCVL2 provide the upper control limits for the CVL values.

Below, we show the outputs for feature selection under the optimal threshold for the Wald tests. The results show that 7 genes (P-value < 0.0205) are selected. Among the 7 genes, two or three genes may be uninformative (FDR $= 0.28 \sim 0.32$).

```
> Wald
$beta
ANXA5       DLG2       ZNF264     DUSP6      CPEB4      LCK          STAT1
-1.0876762  1.3215044  0.5473276  0.7524497  0.5891676  -0.8447389  -0.5844262

$Z
ANXA5      DLG2      ZNF264    DUSP6     CPEB4     LCK         STAT1
-2.885540  2.872880  2.654412  2.628478  2.404015  -2.384028  -2.329287

$P
ANXA5     DLG2      ZNF264    DUSP6     CPEB4     LCK       STAT1
0.003907  0.004067  0.007944  0.008576  0.016216  0.017124  0.019843870

$CVL
CVL          RCVL1        RCVL2
-96.37303    -86.88964    -87.18314

$Genes
No. of genes      No. of selected genes      No. of falsely selected genes
97.000            7.000                      2.265

$FDR
P.value * (No. of genes)        Permutation
0.2840714                       0.3235714
```

Table 1
Six different predictors for overall survival based on the lung cancer data.

	Multigene predictor	CVL	Log-rank	*c*-index
Wald	$(-1.09 \times \text{ANXA5}) + (1.32 \times \text{DLG2}) + (0.55 \times \text{ZNF264}) + (0.75 \times \text{DUSP6}) + (0.59 \times \text{CPEB4}) + (-0.84 \times \text{LCK}) + (-0.58 \times \text{STAT1})$	−96.37	P = 0.452	$c = 0.56$
Score	$(-3.36 \times \text{ANXA5}) + (3.11 \times \text{DLG2}) + (2.81 \times \text{ZNF264}) + (2.71 \times \text{DUSP6}) + (2.53 \times \text{CPEB4}) + (-2.51 \times \text{LCK}) + (-2.45 \times \text{STAT1}) + (2.37 \times \text{STAT2}) + (2.35 \times \text{RNF4}) + (2.23 \times \text{IRF4})$	−95.96	P = 0.664	$c = 0.57$
Copula + Wald	$(0.05 \times \text{ANXA5}) + (0.96 \times \text{DLG2}) + (0.53 \times \text{ZNF264}) + (0.41 \times \text{DUSP6}) + (0.42 \times \text{CPEB4}) + (-0.34 \times \text{LCK}) + (0.01 \times \text{STAT1})$	–	P = 0.189	$c = 0.58$
Copula + score	$(0.05 \times \text{ANXA5}) + (0.96 \times \text{DLG2}) + (0.53 \times \text{ZNF264}) + (0.41 \times \text{DUSP6}) + (0.42 \times \text{CPEB4}) + (-0.34 \times \text{LCK}) + (0.01 \times \text{STAT1}) + (0.43 \times \text{STAT2}) + (0.06 \times \text{RNF4}) + (0.30 \times \text{IRF4})$	–	P = 0.068	$c = 0.58$
Shrinkage + Wald	$(-0.58 \times \text{ANXA5}) + (0.95 \times \text{DLG2}) + (0.20 \times \text{ZNF264}) + (0.55 \times \text{DUSP6}) + (-0.25 \times \text{CPEB4}) + (-0.52 \times \text{LCK}) + (-0.23 \times \text{STAT1})$	–	P = 0.452	$c = 0.57$
Shrinkage + Score	$(-0.61 \times \text{ANXA5}) + (0.89 \times \text{DLG2}) + (0.16 \times \text{ZNF264}) + (0.58 \times \text{DUSP6}) + (-0.36 \times \text{CPEB4}) + (-0.50 \times \text{LCK}) + (-0.13 \times \text{STAT1}) + (0.47 \times \text{STAT2}) + (0.30 \times \text{RNF4}) + (-0.01 \times \text{IRF4})$	–	P = 0.304	$c = 0.60$

CVL, the cross-validated likelihood; **Log-rank**, the P-value of the log-rank test for the difference between good and poor groups in the test samples; ***c*-index**, the concordance measure between a predictor and survival outcome in the test samples.

Below, we show the outputs for feature selection under the optimal threshold for the score tests. The results show that 10 genes (P-value < 0.0275) are selected. Among the 10 genes, three or four genes may be uninformative (FDR $= 0.27 \sim 0.34$).

```
> score
$beta
ANXA5       DLG2       ZNF264      DUSP6      CPEB4       LCK          STAT1
-2.9173315  2.3425963  0.7973673  0.6629881  0.9800451  -0.7055699  -0.9408133
 STAT2       RNF4       IRF4
 1.2701358  0.7446601  0.6878377

$Z
ANXA5     DLG2      ZNF264    DUSP6     CPEB4     LCK         STAT1      STAT2
-3.363578 3.111772  2.814363  2.710854  2.538888  -2.511423  -2.445038  2.369334
RNF4      IRF4
2.345912  2.231286

$P
ANXA5      DLG2       ZNF264     DUSP6      CPEB4      LCK
0.0007693  0.0018596  0.0048874  0.0067110  0.0111205  0.0120245
STAT1      STAT2      RNF4       IRF4
0.0144836  0.0178201  0.0189805  0.0256621

$CVL
CVL           RCVL1         RCVL2
-95.95690     -87.83143     -89.00500

$Genes
No. of genes      No. of selected genes      No. of falsely selected genes
97.000            10.000                     3.405

$FDR
P.value * (No. of genes)          Permutation
0.26675                           0.34050
```

The regression coefficients ($beta) and Z-values ($Z) in the above outputs shall be used to construct a multigene predictor based on the selected genes. Table 1 summarizes the CC predictors constructed by the Wald tests and the score tests.

3.7. Dependent censoring

If censoring is due to informative dropout or any mechanism associated with survival, a predictor calculated from univariate Cox regressions may produce biased results. The method of Emura and Chen [20] aims to adjust the bias by applying a copula [33]. Appendix D provides a short review of copulas and dependent censoring.

A weight in a multigene predictor is computed as $w_j = \hat{\beta}_j(\alpha)$, where α is a copula parameter. In practice, the Clayton copula is used where the parameter α is related to Kendall's tau through $\tau(\alpha) = \alpha/(\alpha + 2)$. Hence, $\alpha = 0$ corresponds to the independent censoring model. The resultant predictor is $\sum_{j \in \Omega} \hat{\beta}_j(\alpha) x_j$, where α is chosen to maximize the predictive capability of the predictor. The *dependCox.reg.CV* function can automatically choose α by maximizing the cross-validated *c*-index; the details are given in Appendix D.

The outputs below show the weights $w_j = \hat{\beta}_j(\hat{\alpha})$ and the estimator $\hat{\alpha} = 11.57$, where the subset Ω consists of the 10 gene as previously selected by the score tests. The input "K = 5" is the number of cross-validation folds for choosing α as recommended in Emura and Chen [21].

```
> dependCox.reg.CV(t.vec,d.vec,X.mat[,names(score$beta)],K=5)
$beta
ANXA5         DLG2          ZNF264        DUSP6         CPEB4
0.053916888   0.959524278   0.534338182   0.411905707   0.418348056
LCK           STAT1         STAT2         RNF4          IRF4
-0.337282371  0.008335039   0.425083915   0.055136154   0.300680681

$SE
ANXA5     DLG2      ZNF264    DUSP6     CPEB4     LCK       STAT1
1.5912224 0.3425241 0.1503290 0.1688951 0.1945337 0.1780180 0.3729820
STAT2     RNF4      IRF4
0.2346972 0.1905754 0.1864976

$Z
ANXA5       DLG2        ZNF264      DUSP6       CPEB4       LCK
0.03388394  2.80133349  3.55445819  2.43882503  2.15051671  -1.89465268
STAT1       STAT2       RNF4        IRF4
0.02234703  1.81120154  0.28931414  1.61224936

$P
ANXA5     DLG2      ZNF264    DUSP6     CPEB4
0.9729697 0.0050892 0.0003787 0.0147351 0.0315143
LCK       STAT1     STAT2     RNF4      IRF4
0.0581384 0.9821711 0.0701097 0.7723409 0.1069076

$alpha    11.57191

$c_index  0.6067678
```

It is interesting to point out that the most significant gene in the univariate Cox model, *ANXA5*, becomes barely significant in the copula-based method. The lack of significance occurs partly due to the concentration of the gene expression values on the category 4:

```
> table(X.mat["ANXA5"])  ## frequency table for the 63 samples ##
 1  3  4
 1  3 59
```

In light of this, estimation of w_j attached to *ANXA5* is inherently difficult. Hence, the weak weight of *ANXA5* derived from the copula-based method is reasonable. A similar phenomenon occurs in *STAT1*.

While the copula-based method has desirable performance, it is computationally very demanding. Therefore, we suggest reducing the number of features before applying the copula-based method as shown in the above example.

3.8. Compound shrinkage predictor

The CS predictor is a refinement of the CC predictor [16]. A weight in the CS predictor is computed as $w_j = \hat{\beta}_j(a)$, where $0 \le a \le 1$ is a shrinkage parameter. The value $a = 0$ yields the CC predictor $\sum_{j\in\Omega} \hat{\beta}_j x_j$, where $\hat{\beta}_j = \arg\max \ell_j(\beta_j)$ is the estimator based on the univariate Cox model. The value $0 < a < 1$ yields the predictor that is intermediate between the univariate $(a = 0)$ and multivariate $(a = 1)$ estimators through a mixture log-likelihood

$$\ell_n^a(\beta) = a\ell_n(\beta) + (1-a)\sum_{j=1}^{p} \ell_j(\beta_j),$$

where $\ell_n(\beta)$ is the log-partial likelihood under a multivariate Cox model. Given a, the maximizer of the mixture log-likelihood is denoted by $\hat{\beta}'(a) = (\hat{\beta}_1(a), \ldots, \hat{\beta}_p(a))$. The resultant predictor is $\sum_{j\in\Omega} \hat{\beta}_j(\hat{a})x_j$ where $\hat{a}$ is chosen to optimize a cross-validated likelihood [16]. The subset Ω may be pre-specified by univariate feature selection.

For instance, one can compute $\hat{\beta}_j(\hat{a})$'s for the 10 genes selected by the score tests, where the estimator $\hat{a} = 0.55$ is chosen by a cross-validation. The input "K = 5″ is the number of cross-validation folds as recommended in [16].

```
> compound.reg(t.vec,d.vec,X.mat[,names(score$beta)],K=5)
$a
[1] 0.55

$beta
ANXA5          DLG2          ZNF264        DUSP6         CPEB4
-0.614820111   0.891692635   0.158262629   0.578653256   -0.357517155
LCK            STAT1         STAT2         RNF4          IRF4
-0.501756022   -0.128728063  0.468369582   0.304780560   -0.005731679
```

3.9. Comparison of multigene predictors

We shall compare the performance of the multigene predictors that just have been introduced. Table 1 summarizes the expressions of the six different multigene predictors that were computed by the 63 training samples.

To evaluate each predictor's test set performance, we applied a multigene predictor to predict survival of the 62 testing samples. We separated the 62 testing samples into either a good prognosis group (low predictor value) or poor prognosis group (high predictor value). We used the median of the predictor values to achieve two equally sized groups [18]. We also computed the *c*-index to measure the concordance between survival and its predictor, which is equivalent to the area under the ROC curve.

Fig. 2 depicts the Kaplan-Meier survival curves for the good (or poor) prognosis group separated by the six different multigene predictors.

The upper parts of Fig. 2 show the results on the CC predictors computed from the optimal Wald tests (upper left) and score tests (upper right). A significant difference between the two groups was not found, but the *c*-index demonstrated a modest ability to predict survival. In the subsequent analyses, we shall examine if the predictive ability is improved by more refined predictors.

The middle parts of Fig. 2 show the results on the copula-based methods. The predictors derived from the copula-based methods improved the ability of discriminating between the good and poor groups over the CC predictors. The modest improvement was also seen in terms of the *c*-index. A clear separation between the good and poor groups was found for the copula-based method applied for the genes optimally selected by the score tests (middle right of Fig. 2), giving a significance level of 10% in the log-rank test.

The bottom parts of Fig. 2 give the results on the CS predictors. The CS predictor improves upon the CC predictor for the genes optimally selected by the score tests. This CS predictor also attains the best value of the *c*-index among the six predictors. However, the log-rank tests performed poorly in this case since it happened that four testing samples had exactly the same predictor values as the median predictor value (ties). Little improvement was seen for the gene optimally selected by the Wald tests.

4. Simulations

We performed two sets of simulations for two different objectives:

Objective (i): to examine the capability of identifying informative genes by using *compound.Cox*.
Objective (ii): to examine the predictive capability of multigene predictors computed by using *compound.Cox*.

Regarding Objective (i), we evaluated the capability of identifying informative genes at the optimal P-value threshold obtained from the CVL plot. This is because one can arbitrary increase the chance of identifying the informative genes by increasing a P-value threshold, which however increases the chance of selecting uninformative genes.

Regarding Objective (ii), we compared the predictive performance of the four predictors: the compound covariate (CC), the compound shrinkage (CS), ridge, and Lasso predictors. That is, we chose the ridge and Lasso predictors as the benchmarks since they are well-known and highly accurate multigene predictors [18,19]. The ridge and Lasso predictors are computed through the R package *penalized* [34].

4.1. Simulation designs

Data were simulated as follows. We generated $p = 5000$ continuously valued gene expressions

$$\mathbf{x}'_i = (\underbrace{x_{i1}, \ldots, x_{i25}}_{\times 25}, \underbrace{x_{i26}, \ldots, x_{i50}}_{\times 25}, \underbrace{x_{i51}, \ldots, x_{i5000}}_{\times (4950)})$$

for n samples via *X.pathway*(n,$p = 5000$,$q1 = 25$,$q2 = 25$) in the *compound.Cox* package. This code generates three independent clusters of gene expressions. The intra-cluster correlation is 0.5 for the first two clusters, and 0 for the last cluster. More details are given in [16]. Let

$$\beta = (\underbrace{\Delta, \ldots, \Delta}_{\times 25}, \underbrace{-\Delta, \ldots, -\Delta}_{\times 25}, \underbrace{0, \ldots, 0}_{\times (4950)}),$$

where $\Delta = 0.1$ for *low signals* and $\Delta = 0.2$ for *high signals*. The first 50 genes are called the *true* genes and the other 4950 genes are called the *false* genes. The goal of Objective (i) is to examine how well *compound.Cox* can select the true genes without selecting the false genes. Given the gene expressions, survival times (T_i) were generated under a Cox model $h(t|\mathbf{x}_i) = h_0(t)\exp(\mathbf{x}'_i\boldsymbol{\beta})$ with $h_0(u) = 1$. Censoring times (U_i) were generated from $U(0,\ 1)$, and $t_i = \min\{T_i, U_i\}$ and $\delta_i = \mathbf{I}\{T_i \le U_i\}$ are computed. The proportion of censored samples is around $\sum_{i=1}^{n}(1-\delta_i)/n \approx 0.56$.

Based on the data $\{(t_i, \delta_i, \mathbf{x}_i);\ i = 1, \ldots, n\}$ generated, we selected genes by the score tests under various P-value thresholds (0.000075∼0.075). We then plotted the CVL values against the P-value thresholds. The CVL values were computed by $K = 5$ fold cross-validation. We also calculated the four multigene predictors based on the selected genes at each threshold, and applied them to the independently simulated test samples. We compared the predictors' performance in terms of the Z-value of the log-rank test for equally sized groups (poor vs. good) and the *c*-index (equivalent to the area under the ROC curve).

Our results are reported on the average of 50 replications. Supplementary Material provides the R codes for simulations.

4.2. Simulation results

Fig. 3 depicts the CVL plot against the P-value threshold for the score tests. Fig. 3 also shows the number of genes selected at the optimum of the CVL plot. Overall, the majority of the true genes were successfully selected while the majority of the false genes are not selected. For instance, 42 genes were selected out of the 50 true genes while only 13 genes were selected out of 4950 false genes under $\Delta = 0.1$ and $n = 100$. Under the large sample size of $n = 200$, the number of successfully selected genes increased and the number of the falsely selected genes reduced. In addition, the accuracy of selected genes under $\Delta = 0.2$ was superior to that under $\Delta = 0.1$. This is because larger signals make it easier to select the true genes. In particular, all the 50 true genes were successfully selected and only 1 false gene were selected under $\Delta = 0.2$ and $n = 200$.

Fig. 4 compares the predictive performance of the four multigene predictors. The four predictors show comparable performance, all exhibiting good ability to predict survival ($|Z\text{-value}| > 6$; *c*-index > 0.8 under $\Delta = 0.1$; $|Z\text{-value}| > 8$; *c*-index > 0.9 under $\Delta = 0.2$). Under $\Delta = 0.2$, the CS predictor performs the best among the four predictors, but it takes the longest computing time. The ridge predictor and the CC predictor are competitive. However, the CC predictor takes the shortest computing time. The Lasso performs relatively poorly, which is expected results in the presence of a large number of informative genes [16].

In summary, our simulation results show that the feature selection methods in *compound.Cox* have a desirable capability of iden-

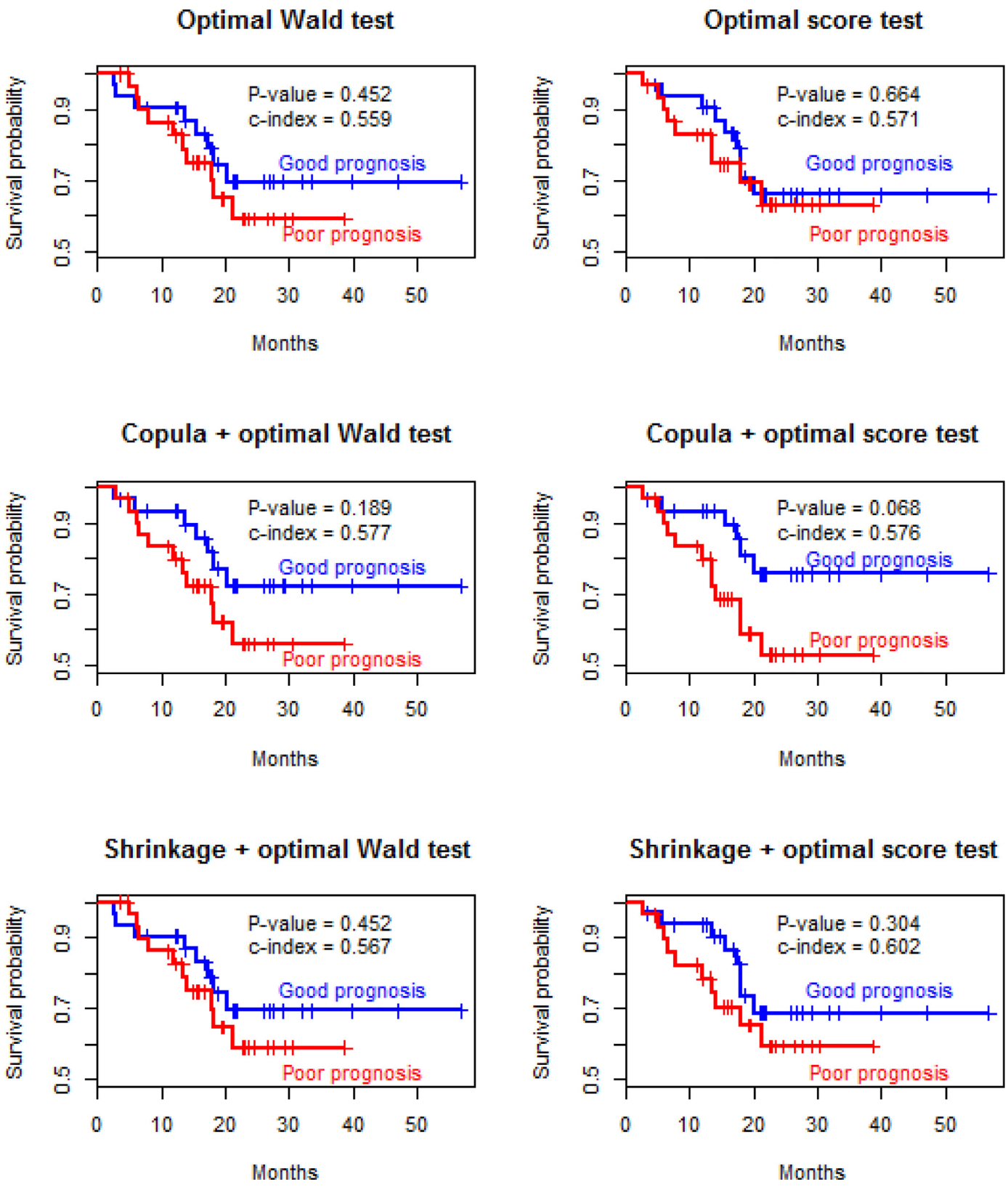
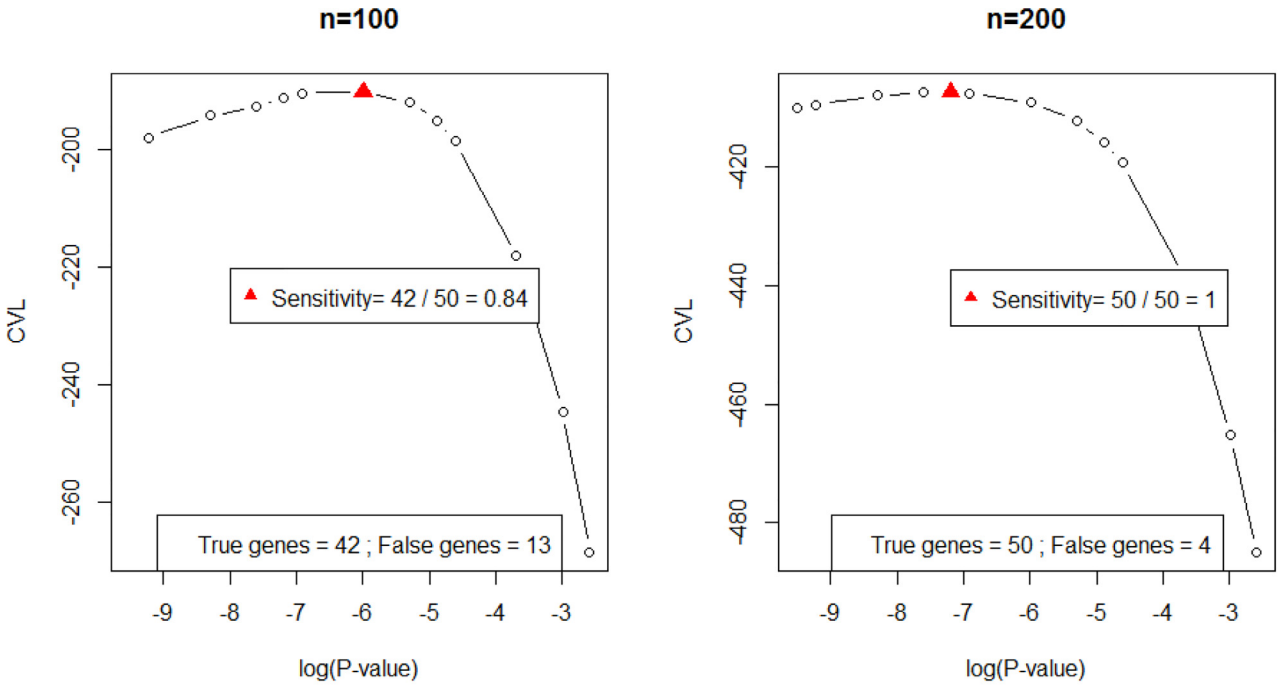
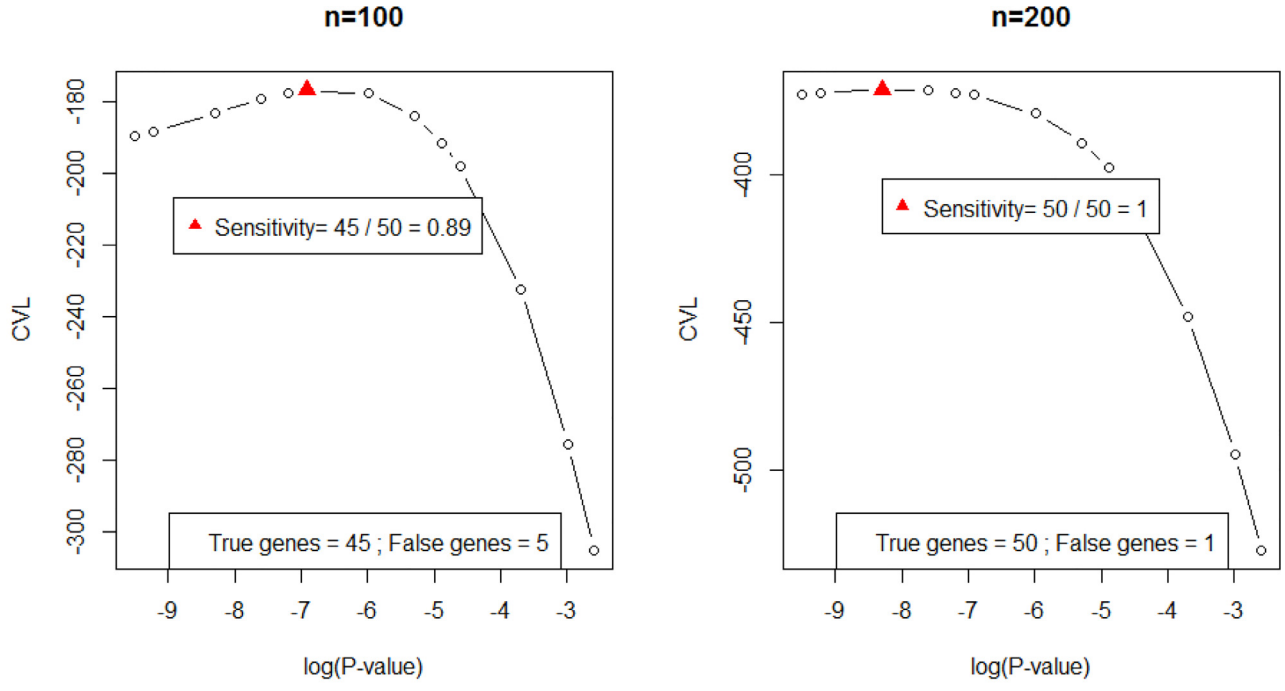


Fig. 2. The Kaplan-Meier curves for the good and poor prognosis groups identified by the six different predictors. The two groups were determined by the low (or high) values of a predictor. The log-rank test was used to measure the difference between the two prognosis groups, where the median of the predictor creates two equally sized groups. The *c*-index was used to measure the concordance between survival and its predictor, which is equivalent to the area under the ROC curve.

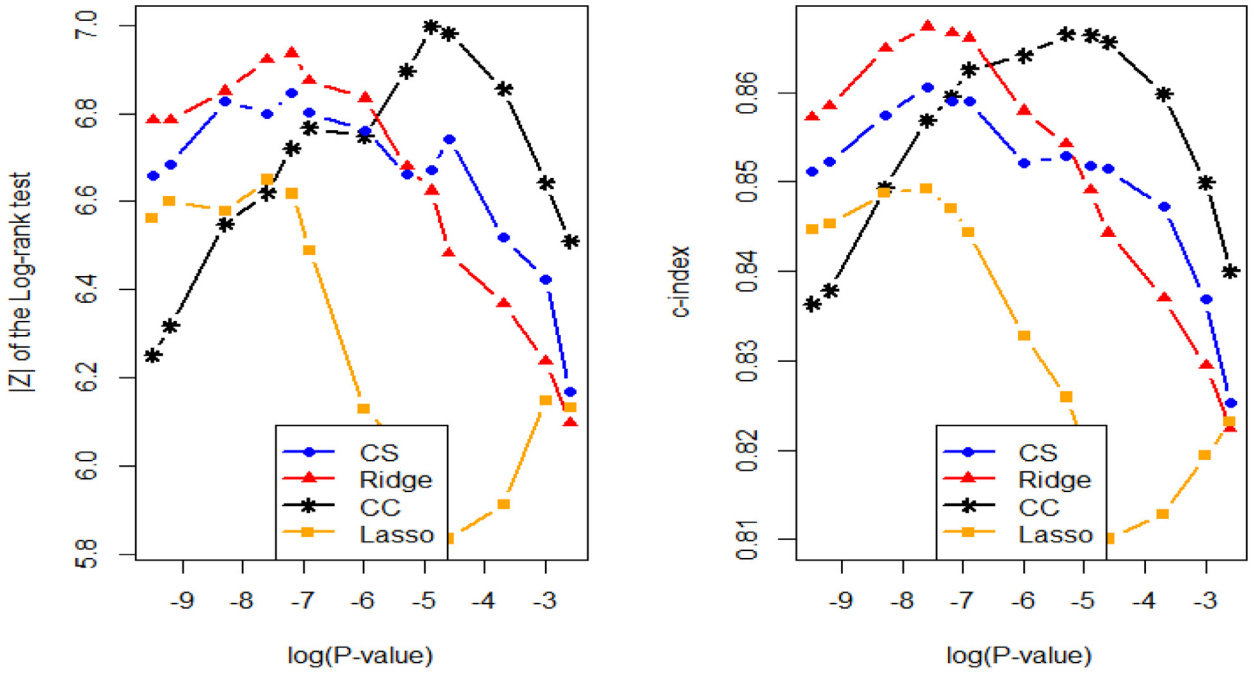


(a) Low signals ($\Delta = 0.1$)

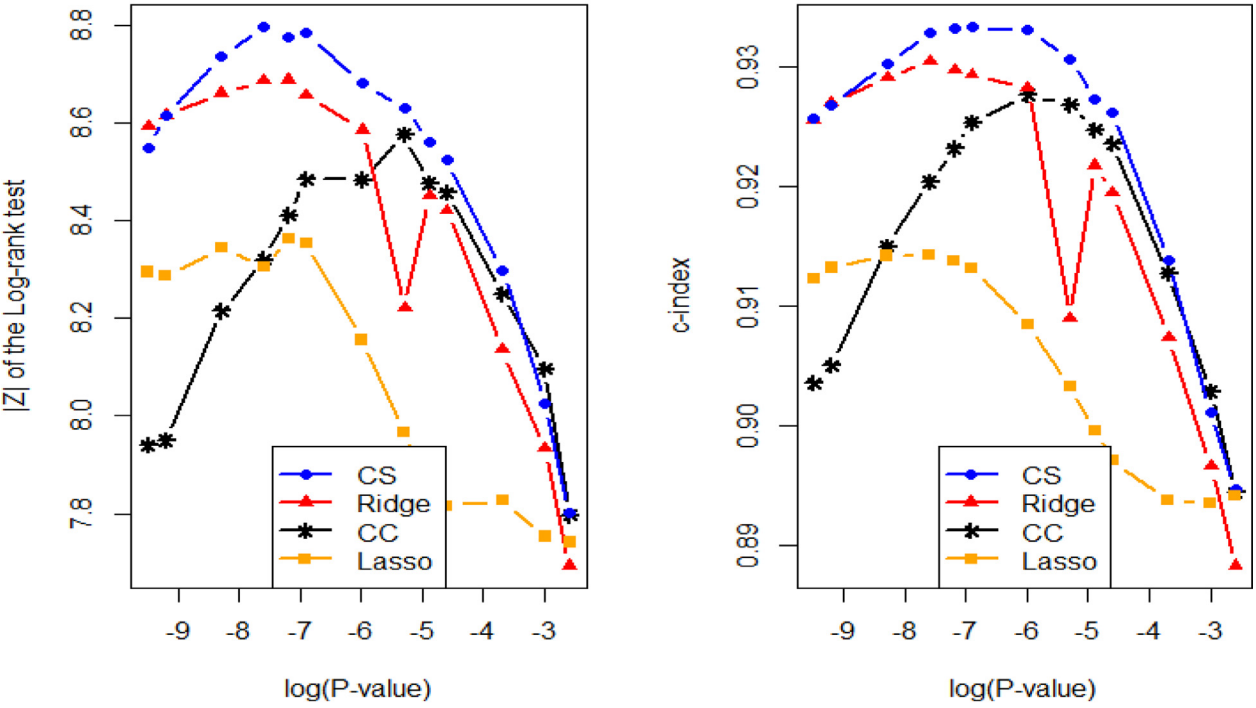


(b) High signals ($\Delta = 0.2$)

Fig. 3. Simulation results for feature selection from 5000 genes (50 true genes + 4950 false genes). The figures report the CVL plots and the number of selected genes at the optimal P-value threshold (▲).



(a) Low signals ($\Delta = 0.1$)



(b) High signals ($\Delta = 0.2$)

Fig. 4. Simulation results for comparing the four mutigene predictors under $n = 100$. High values in the y-axis correspond to better predictive performance.

tifying informative genes. Our simulation results also show that the multigene predictors in *compound.Cox* have competitive performance with the well-established predictors.

5. Discussions

This article demonstrates the ability of the *compound.Cox* package to perform feature selection for predicting survival. We made the lung cancer data available in the package to provide quick and simple illustrations to demonstrate several functions in the package.

Beyond the illustrations for our package, we devised a vector-based computation scheme for the univariate score tests (Section 3.3). This scheme allows us to perform a large number of tests by simple algebras. While the score tests are already known to be computationally more efficient than the Wald tests [1], researchers tend to run each test for each feature repeatedly.

The *compound.Cox* package implements a cross-validation analysis to measure the predictive capability of selected features in terms of the CVL value (Appendix A). The CVL value depends on how to divide the samples into K groups during cross-validation. One could encounter an unusually high CVL value by chance. An obvious strategy to avoid the random variation is to set $K=n$ (leave-one-out cross-validation), or to take the average of CVL values among different sample splits. However, these time-consuming techniques are not recommended. We have proposed a simple quality control method to check if the CVL value is properly computed (also see Fig. 1). This is a procedure newly devised in this article.

We conducted our simulation studies to examine how the truly effective genes are selected by the *compound.Cox* package, where the true relationship between genes and survival is known. Our simulation results confirmed that the package has a high sensitivity to pick up the truly effective genes in the presence of a large number of false genes (Section 4).

The developed algorithms in *compound.Cox* would be useful tools for incorporating high-dimensional features into bivariate survival models, though this topic needs more elaborate analyses. Many bivariate survival models involve frailty [35-37] for heterogeneity or copulas [38,39] for dependence among endpoints. In addition, the issue of dependent censoring arises when an intermediate event (e.g., tumour progression) is observed together with a terminal event (e.g., death) [35,38,40,41]. One pragmatic approach in these complex survival data is to perform univariate feature selection by ignoring dependence between two event times. Our previous work in Emura et al. [11] applied P-value <0.001 threshold to select 128 genes associated with overall survival and 158 genes associated with time-to-tumour progression, yielding the two CC predictors. These CC predictors were then incorporated into a more elaborate bivariate survival model that accounts for heterogeneity and dependence.

The *compound.Cox* package differs remarkably from the existing R packages for feature selection in multivariate Cox regression (*survival* [42]) and penalized Cox regression (*Net-Cox* [43], *SGL* [44], *SIS* [45] and *penalized* [34]). The feature selection methods in the *compound.Cox* package adopt multiple tests with computation of the significance levels of features (in terms of P-value) and the number of false discoveries (in terms of FDR). This is relevant to the objective of achieving biological insights, where screening of prognostic features exhaustively may be a relevant task, even if some selected features are highly correlated. On the other hand, multivariate/penalized Cox regression methods adopt an optimization of a penalized likelihood in term of prediction capability. In some penalized regression, such as Lasso, a feature subset may be identified, taking account of the correlations among features. One has to recognize that such a subset is one, haphazardly selected (due to random errors) from many "solutions" of predictor with comparable predictive capability in high-dimensional situations [46]. The objective of accurate prediction should not be confused with that of achieving biological insight [47,48,17]. It is noted that, compared with penalized regression methods, the *compound.Cox* package can perform well for prediction analysis, as shown in Section 4 (See also Section 1).

Copula-based methods for dealing with dependent censoring also make the *compound.Cox* package different from the existing packages. As shown in the lung cancer data analysis, the copula-based method improves upon the CC predictor; the two Kaplan–Meier survival curves for the good and poor prognosis groups were more clearly separated by the copula-based method (Fig. 2). If dependent censoring exists in the test samples, these Kaplan-Meier estimators could be replaced by the copula-graphic estimators [20,21]. The *compound.Cox* package implements the calculations of the copula-graphic estimator under the Clayton and Gumbel copulas. The details are given in Appendix D.

Acknowledgements

We thank four anonymous reviewers for their helpful comments that greatly improved the manuscript. Emura T is financially supported by Ministry of Science and Technology, Taiwan (103-2118-M-008-MY2; 107-2118-M-008-003-MY3). Matsui S is financially supported by a Grant-in-Aid for Scientific Research (16H06299) and JST-CREST (JPMJCR1412) from the Ministry of Education, Culture, Sports, Science and Technology of Japan. Mathematics in Biology Group of Institute of Statistical Science, Academia Sinica, supported data collection.

Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.cmpb.2018.10.020.

Appendix A: Cross-validated likelihood (CVL)

We introduce the CVL proposed by Matsui [6] because its definition is complex and not sufficiently detailed in his original article. In addition, we propose a quality control method to check if the CVL value is properly calculated.

Let $\{(t_i, \delta_i, \mathbf{x}_i);\ i=1,\ \ldots,\ n\}$ be survival data. To perform a K-fold cross validation, we first divide the n samples into K groups of approximately equal sample sizes and label them as $\Im_k$ for $k=1,\ \ldots,\ K$ such that $\cup_{k=1}^{K}\Im_k=\{1,\ \ldots,\ n\}$. Define $\Im_{-k}$ as a training set for a test set $\Im_k$ such that $\Im_k\cup\Im_{-k}=\{1,\ \ldots,\ n\}$, $k=1,\ \ldots,\ K$. The CC predictor based on the training set $\Im_{-k}$ is denoted by $\mathrm{CC}_{i,-k}=\sum_{j\in\Omega_{-k}} w_{j,-k}x_{ij}$, where the set Ω_{-k} and the weight $w_{j,-k}$ are determined by the training set $\Im_{-k}$ given a P-value threshold.

Treating $\mathrm{CC}_{i,-k}$ as a covariate for (t_i, δ_i), we have a new set of survival data $\{(t_i, \delta_i, \mathrm{CC}_{i,-k});\ i\in\Im_{-k}\}$. We fit the data to a Cox model

$h(t|\mathrm{CC}_{\ell,-k})=h_0(t)\exp(\gamma\mathrm{CC}_{\ell,-k})$, and obtain an estimator $\hat{\gamma}_{-k}$. The CVL is defined as

$$CVL=\sum_{k=1}^{K}\{\ell(\hat{\gamma}_{-k})-\ell_{-k}(\hat{\gamma}_{-k})\}, \tag{1}$$

where $\hat{\gamma}_{-k}=\arg\max_{\gamma}\ell_{-k}(\gamma)$,

$$\ell(\gamma)=\sum_i \delta_i\left[\gamma\mathrm{CC}_{i,-k}-\log\left\{\sum_{\ell\in R_i}\exp\left(\gamma\mathrm{CC}_{\ell,-k}\right)\right\}\right], \tag{2}$$

$$\ell_{-k}(\gamma)=\sum_{i\in\Im_{-k}} \delta_i\left[\gamma\mathrm{CC}_{i,-k}-\log\left\{\sum_{\ell\in R_i\cap\Im_{-k}}\exp\left(\gamma\mathrm{CC}_{\ell,-k}\right)\right\}\right], \tag{3}$$

$$\begin{bmatrix} t_1, \delta_1, x_{11}, x_{12}, \ldots\ldots\ldots\ldots, x_{1p} \\ \vdots \\ t_n, \delta_n, x_{n1}, x_{n2}, \ldots\ldots\ldots\ldots, x_{np} \end{bmatrix} : \text{survival data with } n \text{ samples and } p \text{ genes}$$

$\Downarrow$ ***Step 1***: Randomly divide n samples between testing samples and training samples

$$\left.\begin{bmatrix} t_1, \delta_1, x_{11}, x_{12}, \ldots\ldots\ldots\ldots, x_{1p} \\ \vdots \end{bmatrix}\right\} \quad \frac{n}{K} \text{ testing samples}, \quad i \in \Im_k$$

$$\begin{bmatrix} \text{------} \\ \vdots \\ t_n, \delta_n, x_{n1}, x_{n2}, \ldots\ldots\ldots\ldots, x_{np} \end{bmatrix}\Bigg\} \quad n - \frac{n}{K} \text{ training samples}, \quad i \in \Im_{-k}$$

$\Downarrow$ ***Step 2***: Select genes using the training samples only, resulting in the reduced list of genes ($q = q_k$ genes with P-values < P, where P is prespecified)

$$\begin{bmatrix} t_1, \delta_1, x_{11}, x_{12}, \ldots, x_{1q} \\ \vdots \\ t_m, \delta_m, x_{m1}, x_{m2}, \ldots, x_{mq} \end{bmatrix}, \quad m = n - \frac{n}{K}$$

$\Downarrow$ ***Step 3***: Calculate CCs using the weights computed by the training samples

$$\begin{bmatrix} t_1, \delta_1, \mathrm{CC}_{1,-k} \\ \vdots \\ t_m, \delta_m, \mathrm{CC}_{m,-k} \end{bmatrix}, \quad \mathrm{CC}_{i,-k} = w_{1,-k}x_{i1} + w_{2,-k}x_{i2} + \cdots + w_{q,-k}x_{iq}, \quad i \in \Im_{-k}$$

$\Downarrow$ ***Step 4***: Perform likelihood analysis based solely on the training samples

$$\ell_{-k}(\gamma) = \sum_{i \in \Im_{-k}} \delta_i \left[\gamma\, \mathrm{CC}_{i,-k} - \log\left\{ \sum_{\ell \in R_i \cap \Im_{-k}} \exp(\gamma\, \mathrm{CC}_{\ell,-k}) \right\} \right], \quad \hat{\gamma}_{-k} = \arg\max_{\gamma} \ell_{-k}(\gamma)$$

$\Downarrow$ ***Step 5***: Repeat ***Step 2 - Step 4***:for $k = 1, 2, \cdots, K$, and compute

$$CVL = \sum_{k=1}^{K} \{ \ell(\hat{\gamma}_{-k}) - \ell_{-k}(\hat{\gamma}_{-k}) \}, \qquad \ell(\gamma) = \sum_i \delta_i \left[\gamma \mathrm{CC}_{i,-k} - \log\left\{ \sum_{\ell \in R_i} \exp\left(\gamma \mathrm{CC}_{\ell,-k}\right) \right\} \right]$$

Fig. A1. The algorithm for calculating the cross-validated likelihood (CVL).

where $R_i = \{ \ell\colon t_\ell \geq t_i \}$ is the risk set. The computation scheme of the CVL is given in Fig. A1.

We propose a quality control method to check if the CVL value is proper. This is because the CVL value is subject to some variation due to random allocation of the samples into K groups. For this purpose, we shall introduce the RCVL1 and the RCVL2 values that provide upper control limits for the CVL values. If the CVL value is less than the RCVL1 and RCVL2 values, the CVL value would be in-control. On the other hand, if the CVL value exceeds either the RCVL1 or RCVL2 value, then the CVL may be computed again after changing the sample allocation.

RCVL1 (re-substitution CVL without cross-validation):

We define the RCVL1 value by Eqs. (1)–(3), where we replace $\mathrm{CC}_{i,-k} = \sum_{j \in \Omega_{-k}} w_{j,-k}x_{ij}$ by $\mathrm{CC}_i = \sum_{j \in \Omega} w_j x_{ij}$, where the set Ω and the weight w_j are determined by the whole samples. Since the RCVL1 value is a re-substitution estimate, it gives an upward bias relative to the CVL value [24]. Specifically, the information of the i-th sample is incorporated into the weight w_j and set Ω, violating the principle of cross-validation. Usually, the RCVL1 value increases as the number of features in Ω increases. Since the RCVL1 value is more robust against the sample allocation, it serves as an upper control limit for the CVL value.

RCVL2 (re-substitution CVL with incomplete cross-validation)

We define the RCVL2 value by Eqs. (1)-(3), where we replace $\mathrm{CC}_{i,-k} = \sum_{j \in \Omega_{-k}} w_{j,-k}x_{ij}$ by $\mathrm{CC}_{i,-k} = \sum_{j \in \Omega} w_{j,-k}x_{ij}$, where the set Ω is determined by the whole samples, but the weight $w_{j,-k}$ is determined by the training data $\Im_{-k}$. Since the RCVL2 value is a re-substitution estimate, it gives an upward bias relative to the CVL value [24]. Specifically, the information of the i-th sample is incorporated into the set Ω, violating the principle of cross-validation. Since the RCVL2 value is more robust against the sample allocation, it serves as an upper control limit for the CVL value.

Appendix B: False discovery rate (FDR)

Performing a number of multiple tests often leads us to evaluate the number of falsely rejected hypotheses. For instance, if P-value < 0.001 is a criterion for rejecting hypotheses, the tests for $p = 5{,}000$ features would identify 5 false features by chance. The false discovery rate (FDR) is the percentage of such false features.

As in Witten and Tibshirani [1], we suggested the following permutation method to compute the FDR.

- Step 1: Randomly generate M permutations: $\{ \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \} \to \{ \mathbf{x}_1^m, \mathbf{x}_2^m, \ldots, \mathbf{x}_n^m \}$ for $m = 1, \ldots, M$. Obtain permuted samples $\{ (t_i, \delta_i, \mathbf{x}_i^{(m)});\ i = 1, \ldots, n \}$ for $m = 1, \ldots, M$.
- Step 2: For each m, perform feature selection (via the Wald tests or score tests), and the selected features are represented as a set $\Omega^{(m)} = \{ j : P_j^{(m)} < P \}$, where $P_j^{(m)}$ is the P-value for testing H_{0j}: $\beta_j = 0$ vs. H_{1j}: $\beta_j \neq 0$ based on the permuted samples $\{ (t_i, \delta_i, \mathbf{x}_i^{(m)});\ i = 1, \ldots, n \}$.

- Step 3: Compute the ratio

$$\text{FDR} = \frac{\text{The expected number of false discoveries}}{\text{The number of rejections}} = \frac{\frac{1}{M}\sum_{m=1}^{M}\sum_{j=1}^{p} I(P_j^{(m)} < P)}{\sum_{j=1}^{p} I(P_j < P)}.$$

Alternatively, one may compute the FDR by the simple formula

$$\text{FDR} = \frac{\text{The expected number of false discoveries}}{\text{The number of rejections}} = \frac{P \times p}{q},$$

where P is the P-value threshold (e.g., $P=0.05$), p is the total number of features (tests), and q is the number of selected features. The latter formula relies on the assumption that each test statistic has the correct null distribution, namely $\Pr(P_j < P) = P$ under the null hypothesis. Under this assumption, it can be shown that

$$\frac{\frac{1}{M}\sum_{m=1}^{M}\sum_{j=1}^{p} I(P_j^{(m)} < P)}{\sum_{j=1}^{p} I(P_j < P)} \approx \frac{E\left[\sum_{j=1}^{p} I(P_j^{(m)} < P)\right]}{q} \approx \frac{p \times E[I(P_j^{(m)} < P)]}{q} = \frac{p \times P}{q}.$$

Appendix C: Outputs for feature selection by score tests

```
> uni.selection(t.vec,d.vec,X.mat,K=20,P.value=0.05,score=TRUE,permutation=TRUE) ## Score test
$beta
ANXA5       DLG2       ZNF264     DUSP6      CPEB4      LCK         STAT1
-2.9173315  2.3425963  0.7973673  0.6629881  0.9800451  -0.7055699  -0.9408133
STAT2       RNF4       IRF4       NF1        HGF        ERBB3       MMD
1.2701358   0.7446601  0.6878377  0.7960428  0.7376547  0.5036743   0.8178961
FRAP1       KIDINS220  HMMR       TAL1
-0.7369568  1.5165640  0.5966055  -0.5482119

$Z
ANXA5      DLG2      ZNF264    DUSP6     CPEB4     LCK        STAT1      STAT2
-3.363578  3.111772  2.814363  2.710854  2.538888  -2.511423  -2.445038  2.369334
RNF4       IRF4      NF1       HGF       ERBB3     MMD        FRAP1      KIDINS220
2.345912   2.231286  2.195686  2.193421  2.171900  2.086149   -2.072383  2.052683
HMMR       TAL1
1.988259   -1.974660

$P
ANXA5          DLG2           ZNF264         DUSP6          CPEB4          LCK
0.0007693912   0.0018596776   0.0048874068   0.0067110076   0.0111205531   0.0120245448
STAT1          STAT2          RNF4           IRF4           NF1            HGF
0.0144836682   0.0178201514   0.0189805779   0.0256621824   0.0281144323   0.0282770689
ERBB3          MMD            FRAP1          KIDINS220      HMMR           TAL1
0.0298632024   0.0369651109   0.0382297696   0.0401033407   0.0467830229   0.0483067594

$CVL
CVL          RCVL1        RCVL2
-97.36951    -88.13196    -91.71561

$Genes
No. of genes     No. of selected genes   No. of falsely selected genes
97.000           18.000                  5.545

$FDR
P.value * (No. of genes)     Permutation
0.2694444                    0.3080556
```

Appendix D: Copula-based methods for dependent censoring

We describe the copula-based methods for dependent censoring. More details can be found in the book of Emura and Chen [21]. We also explain how the *compound.Cox* package implements the copula-based methods.

Consider *random variables*,

- T: survival time
- U: censoring time

Consider a copula model for dependent censoring

$\Pr(T > t, U > u) = C_\alpha(S_T(t), S_U(u))$, where C_α is a copula function; $S_T(t) = \Pr(T > t)$ and $S_U(u) = \Pr(U > u)$ are the marginal survival functions. Some well-known copulas are

The independence copula:

$$C(u, v) = uv,$$

The Clayton copula:

$$C_\alpha(u, v) = (u^{-\alpha} + v^{-\alpha} - 1)^{-1/\alpha}, \quad \alpha > 0,$$

The Gumbel copula:

$$C_\alpha(u, v) = \exp[-\{(-\log u)^{\alpha+1} + (-\log v)^{\alpha+1}\}^{\frac{1}{\alpha+1}}], \quad \alpha \geq 0,$$

The parameter α represents the measure of dependence, and can be transformed to Kendall's tau (τ). For instance, it can be show that $\tau = \alpha/(\alpha+2)$ under the Clayton copula and that $\tau = \alpha/(\alpha+1)$ under the Gumbel copula.

Let (t_i, δ_i), $i = 1, \ldots, n$, be survival data without covariates, where $t_i = \min\{T_i, U_i\}$, $\delta_i = \mathbf{I}(T_i \le U_i)$, and $\mathbf{I}(\cdot)$ is the indicator function. Assume that all the observed times are distinct ($t_i \neq t_j$ whenever $i \neq j$). Let $n_i = \sum_{\ell=1}^{n} \mathbf{I}(t_\ell \ge t_i)$ be the *number at-risk* at time t_i.

Under the Clayton copula, the copula-graphic estimator is computed as

$$\hat{S}_T(t) = \left[1 + \sum_{t_i \le t,\ \delta_i = 1} \left\{ \left(\frac{n_i - 1}{n}\right)^{-\alpha} - \left(\frac{n_i}{n}\right)^{-\alpha} \right\} \right]^{-1/\alpha}$$

Under the Gumbel copula, the copula-graphic estimator is computed as

$$\hat{S}_T(t) = \exp\left(-\left[\sum_{t_i \le t,\ \delta_i = 1} \left\{ -\log\left(\frac{n_i - 1}{n}\right) \right\}^{\alpha+1} - \left\{ -\log\left(\frac{n_i}{n}\right) \right\}^{\alpha+1} \right]^{\frac{1}{\alpha+1}} \right)$$

These two copula-graphic estimators can be computed by *CG.Clayton()* and *CG.Gumbel()*. Under the independence copula, the copula-graphic estimator is equal to the Kaplan-Meier estimator.

Let $\{(t_i, \delta_i, x_{ij});\ i = 1, \ldots, n\}$ be survival data for the j-th feature ($j = 1, \ldots, p$). The data are fitted to the copula model

$$\Pr(T > t,\ U > u | x_j) = C_\alpha\{\Pr(T > t | x_j),\ \Pr(U > u | x_j)\},$$

where $\Pr(T > t | x_j) = \exp\{-\Lambda_{0j}(t)\exp(\beta_j x_j)\}$, $\Pr(U > u | x_j) = \exp\{-\Gamma_{0j}(u)\exp(\gamma_j x_j)\}$, and the copula C_α is the same for every j. For a given value of α, the semiparametric maximum likelihood estimator $(\hat{\beta}_j(\alpha),\ \hat{\gamma}_j(\alpha),\ \hat{\Lambda}_{0j}(\alpha),\ \hat{\Gamma}_{0j}(\alpha))$ can be calculated by using *dependCox.reg()*. Repeating this calculation for $j = 1, \ldots, p$, the vector $(\hat{\beta}_1(\alpha), \ldots, \hat{\beta}_p(\alpha))$ is obtained. Here, the value of α can be chosen by maximizing a cross-validated c-index, which is a concordance measure between the outcome (t_i, δ_i) and its predictor $\sum_{j \in \Omega} \hat{\beta}_j^{(-i)}(\alpha) x_j$ [20,21]. One can simply use *dependCox.reg.CV()* to calculate $(\hat{\beta}_1(\hat{\alpha}), \ldots, \hat{\beta}_p(\hat{\alpha}))$.

References

[1] D.M. Witten, R. Tibshirani, Survival analysis with high-dimensional covariates, Stat. Methods Med. Res. 19 (2010) 29–51.
[2] D.G. Beer, S.L.R. Kardia, Huang CC., Giordano TJ, Levin AM, et al. Gene-expression profiles predict survival of patients with lung adenocarcinoma, Nature Medicine 8 (2002) 816–824.
[3] A. Rosenwald, G. Wright, W.C. Chan, J.M. Connors, E. Campo, et al., The use of molecular profiling to predict survival after chemotherapy for diffuse large-B–cell lymphoma, N. Engl. J. Med. 346 (25) (2002) 1937–1947.
[4] J.R. Vasselli, J.H. Shih, S.R. Iyengar, J. Maranchie, J. Riss, et al., Predicting survival in patients with metastatic kidney cancer by gene-expression profiling in the primary tumor, Proc. Natl. Acad. Sci. 100 (12) (2003) 6958–6963.
[5] Y. Wang, J.G. Klijn, Y. Zhang, A.M. Sieuwerts, et al., Gene-expression profiles to predict distant metastasis of lymph-node-negative primary breast cancer, Lancet 365 (9460) (2005) 671–679.
[6] S. Matsui, Predicting survival outcomes using subsets of significant genes in prognostic marker studies with microarrays, BMC Bioinformatics 7 (2006) 156.
[7] H.Y. Chen, S.L. Yu, C.H. Chen, G.C. Chang, C.Y. Chen, et al., A five-gene signature and clinical outcome in non-small-cell lung cancer, N. Engl. J. Med. 356 (2007) 11–20.
[8] C. Yau, L. Esserman, D.H. Moore, Sninsky F.J. Waldman, C.C. BenzE, A multigene predictor of metastatic outcome in early stage hormone receptor-negative and triple-negative breast cancer, BMC Breast Cancer Res. 12 (2010) R85.
[9] K. Yoshihara, A. Tajima, T. Yahata, S. Kodama, H. Fujiwara, et al., Gene expression profile for predicting survival in advanced-stage serous ovarian cancer across two independent datasets, PloS One 5 (3) (2010) e9615.
[10] S. Matsui, R.M. Simon, P. Qu, J.D. Shaughnessy, B. Barlogie, J. Crowley, Developing and validating continuous genomic signatures in randomized clinical trials for predictive medicine, Clinical Cancer Res. 18 (21) (2012) 6065–6073.
[11] T. Emura, M. Nakatochi, S. Matsui, H. Michimae, V. Rondeau, Personalized dynamic prediction of death according to tumour progression and high-dimensional genetic factors: meta-analysis with a joint model, Stat. Method Med. Res. 27 (9) (2018) 2842–2858.
[12] C. Lai, M.J. Reinders, L.J. van't Veer, L.F. Wessels, A comparison of univariate and multivariate gene selection techniques for classification of cancer datasets, BMC Bioinform. 7 (2006) 235.
[13] S.D. Zhao, G. Parmigiani, C. Huttenhower, L. Waldron, Más-o-menos: a simple sign averaging method for discrimination in genomic data analysis, Bioinformatics 30 (21) (2014) 3062–3069.
[14] L. Waldron, B. Haibe-Kains, A.C. Culhane, M. Riester, J. Ding, et al., Comparative meta-analysis of prognostic gene signatures for late-stage ovarian cancer, J. Natl. Cancer Inst. 106 (5) (2014) dju049.
[15] S. Dudoit, J. Fridlyand, T.P. Speed, Comparison of discrimination methods for the classification of tumors using gene expression data, J Am. Stat. Assoc. 97 (2002) 77–87.
[16] T. Emura, Y.H. Chen, H.Y. Chen, Survival prediction based on compound covariate under Cox proportional hazard models, PLoS One 7 (10) (2012) e47627, doi:10.1371/journal.pone.0047627.
[17] S. Matsui, Statistical issues in clinical development and validation of genomic signatures, in: S Matsui, M Buyse, R. Simon (Eds.), Design and Analysis of Clinical Trials For Predictive Medicine, CRC Press, Boca Raton, 2015, pp. 207–226.
[18] H.M. Bøvelstad, S. Nygård, H.L. Storvold, M. Aldrin, Borgan Ø, et al. Predicting survival from microarray data – a comparative study, Bioinformatics 23 (2007) 2080–2087.
[19] W.N. van Wieringen, D. Kun, R. Hampel, A.L. Boulesteix, Survival prediction using gene expression data: A review and comparison, Comp. Stat. Data Anal. 53: (2009) 1590–1603.
[20] T. Emura, Y.H. Chen, Gene selection for survival data under dependent censoring, a copula-based approach, Statist. Method Med. Res. 25 (6) (2016) 2840–2857.
[21] T. Emura, Y.H. Chen, Analysis of Survival Data with Dependent Censoring, Copula-Based Approaches, JSS Research Series in Statistics, Springer, Singapore, 2018.
[22] T. Emura, H.Y. Chen, S. Matsui, Y.H. Chen, compound.Cox: univariate feature selection and compound covariate for predicting survival, CRAN (2018) Version 3.14.
[23] R. Simon, Design and Analysis of DNA Microarray Investigations, Springer, New York, 2003.
[24] R. Simon, Roadmap for developing and validating therapeutically relevant genomic classifiers, J. Clin. Oncol. 23 (29) (2005) 7332–7341.
[25] J.W. Tukey, Tightening the clinical trial, Controlled Clinical Trials 14 (1993) 266–285.
[26] I.S. Lossos, D.K. Czerwinski, A.A. Alizadeh, M.A. Wechser, R. Tibshirani, D. Botstein, R. Levy, Prediction of survival in diffuse large-B-cell lymphoma based on the expression of six genes, N. Engl. J. Med. 350 (18) (2004) 1828–1837.
[27] D. Collett, 3rd edition, Modelling Survival Data in Medical Research, CRC press, London, 2015.
[28] L.P. Rivest, M.T. Wells, A martingale approach to the copula-graphic estimator for the survival function under dependent censoring, J. Multivar. Anal. 79 (2001) 138–155.
[29] T. Emura, H. Michimae, A copula-based inference to piecewise exponential models under dependent censoring, with application to time to metamorphosis of salamander larvae, Environ. Ecol. Stat. 24 (1) (2017) 151–173.
[30] Y.H. Chen, Semiparametric marginal regression analysis for dependent competing risks under an assumed copula, J. R. Stat. Soc., Ser. B 72 (2010) 235–251.
[31] N.D. Staplin, A.C. Kimber, D. Collett, P.J. Roderick, Dependent censoring in piecewise exponential survival models, Stat. Methods Med. Res. 24 (3) (2015) 325–341.
[32] H. Moradian, D. Denis Larocque, F. Bellavance, Survival forests for data with dependent censoring, Stat. Method Med. Res. (2017), doi:10.1177/0962280217727314.
[33] R.B. Nelsen, An Introduction to Copulas, 2nd Edition, Springer, New York, 2006.
[34] J. Goeman, R. Meijer, N. Chaturvedi, M. Lueder, penalized: L1 (Lasso and Fused Lasso) and L2 (ridge) penalized estimation in GLMs and in the Cox model, CRAN (2017) Version 0.9-50.
[35] V. Rondeau, J.P. Pignon, S. Michiels, A joint model for dependence between clustered times to tumour progression and deaths: A meta-analysis of chemotherapy in head and neck cancer, Statist. Methods Med. Res. 24 (6) (2015) 711–729.
[36] V. Rondeau, J.R. Gonzalez, frailtypack: A computer program for the analysis of correlated failure time data using penalized likelihood estimation, Comput. Methods Prog. Biomed. 80 (2) (2005) 154–164.
[37] I.D. Ha, N.J. Christian, J.H. Jeong, J. Park, Y. Lee, Analysis of clustered competing risks data using subdistribution hazard models with multivariate frailties, Stat. Methods Med. Res. 25 (6) (2016) 2488–2505.
[38] T. Emura, M. Nakatochi, K. Murotani, V. Rondeau, A joint frailty-copula model between tumour progression and death for meta-analysis, Stat. Methods Med. Res. 26 (6) (2017) 2649–2666.
[39] M. Peng, L. Xiang, S. Wang, Semiparametric regression analysis of clustered survival data with semi-competing risks, Comp. Stat. Data Anal. 124 (2018) 53–70.
[40] J.P. Fine, H. Jiang, R. Chappell, On semi-competing risks data, Biometrika 88 (2001) 907–920.
[41] S. Haneuse, K.H. Lee, Semi-competing risks data analysis, accounting for death as a competing risk when the outcome of interest is nonterminal, Circ. Cardiovasc. Qual. Outcomes 9 (2016) 322–331.
[42] T.M. Therneau, T. Lumley, survival: survival analysis, CRAN (2017) Version 2.41-3.

[43] W. Zhang, T. Ota, V. Shridhar, J. Chien, B. Wu, R. Kuang, Network-based survival analysis reveals subnetwork signatures for predicting outcomes of ovarian cancer treatment, PLoS Comput. Biol. 9 (3) (2013) e1002975.
[44] N. Simon, J. Friedman, T. Hastie, R. Tibshirani, SGL: Fit a GLM (or cox model) with a combination of lasso and group lasso regularization, CRAN (2013) Version 1.1.
[45] D.F. Saldana, Y. Feng, SIS: An R package for sure independence screening in ultrahigh dimensional statistical models, J. Stat. Softw. 83 (2) (2018) 1–25.
[46] M. Schumacher, N. Hollander, G. Schwarzer, H. Binder, W. Sauerbrei, Prognostic factor studies., in: J.J. Crowley, A. Hoering (Eds.), Handbook of Statistics in Clinical Oncology, 3rd edition, CRC Press, Boca Raton, 2012, pp. 415–469.
[47] S.L. George, Statistical issues in translational cancer research, Clin. Cancer Res. 14 (2008) 5954–5958.
[48] R. Simon, The use of genomics in clinical trial design, Clin. Cancer Res. 14 (2008) 5984–5993.

Supplementary Material

compound.Cox: univariate feature selection and compound covariate for predicting survival, *Computer Methods and Programs in Biomedicine*

Corresponding to Takeshi Emura (takeshiemura@gmail.com)

S1. R codes for the analysis of the lung cancer data:

The R codes perform the following tasks:

- Display "Lung" containing 125 samples (63 training samples + 62 test samples).
- Perform univariate feature selection using the 63 training samples.
- Draw the CVL plot for identifying the optimal P-value threshold.
- Compute six predictors and apply them to separate the test samples into a good or poor prognosis group.
- Compare the two Kaplan-Meier survival curves in the test samples (good vs. poor).

```
library(compound.Cox)
data("Lung")
Lung

train=Lung$train ## index for training samples

### 63 training samples ###
t.vec=Lung$t.vec[train]
d.vec=Lung$d.vec[train]
X.mat=as.matrix(Lung[,-c(1,2,3)][train,])

### Feature selection (P-value < 0.05) ###
uni.selection(t.vec,d.vec,X.mat,K=20,P.value=0.05,score=FALSE,permutation=TRUE) ## Wald test
uni.selection(t.vec,d.vec,X.mat,K=20,P.value=0.05,score=TRUE,permutation=TRUE) ## Score test

#########################################################
#    Draw the CVL plot to find the optimal P-value threshold     #
#########################################################
P.grid=seq(0.01,0.08,length=21) ## 0.01<P-value<0.08 ##
K=20 ## 20-fold cross-validation ##
CVL.score=CVL.Wald=NULL
RCVL1.score=RCVL1.Wald=NULL
RCVL2.score=RCVL2.Wald=NULL

for(P in P.grid){
```

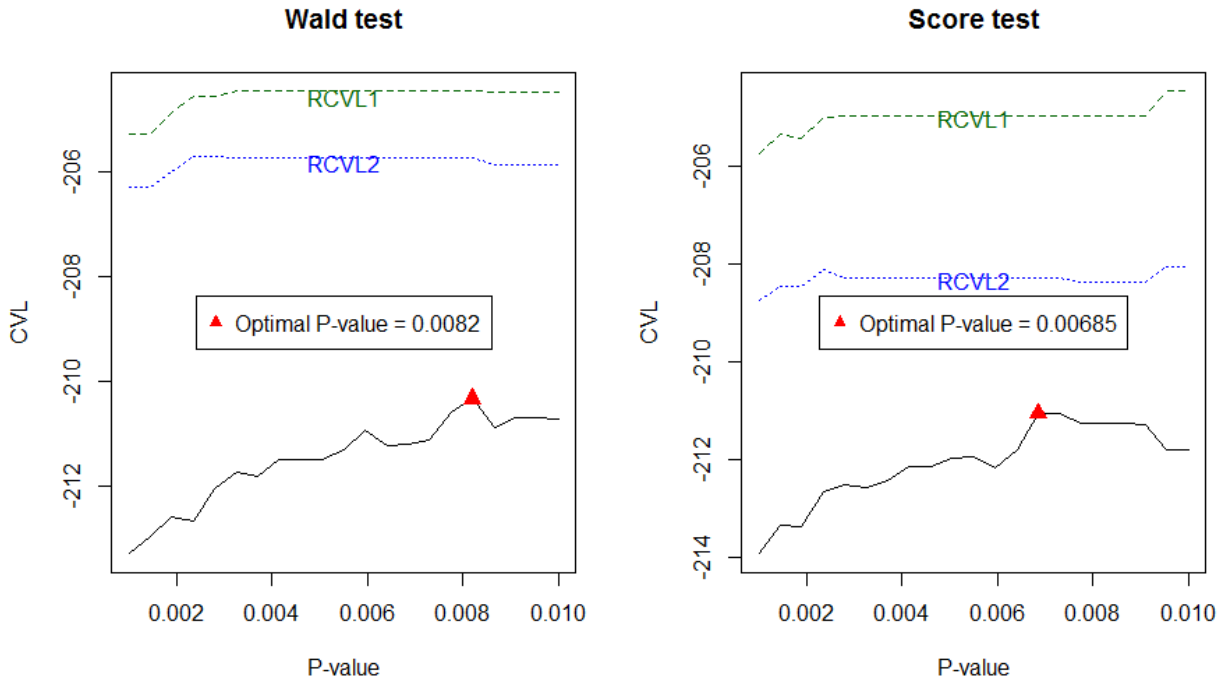
```
  res.score=uni.selection(t.vec,d.vec,X.mat,K=K,P.value=P,score=TRUE)
  res.Wald=uni.selection(t.vec,d.vec,X.mat,K=K,P.value=P,score=FALSE)
  CVL.score=c(CVL.score,res.score$CVL[1])
  CVL.Wald=c(CVL.Wald,res.Wald$CVL[1])
  RCVL1.score=c(RCVL1.score,res.score$CVL[2])
  RCVL1.Wald=c(RCVL1.Wald,res.Wald$CVL[2])
  RCVL2.score=c(RCVL2.score,res.score$CVL[3])
  RCVL2.Wald=c(RCVL2.Wald,res.Wald$CVL[3])
}
P.Wald=P.grid[which.max(CVL.Wald)]
P.score=P.grid[which.max(CVL.score)]

######## The plot of CVL (Wald tests)  #########
Range.Wald= range(c(CVL.Wald,RCVL1.Wald,RCVL2.Wald),na.rm=TRUE)
par(mfrow=c(1,2))
plot(P.grid,CVL.Wald,type="l",ylim=Range.Wald,xlab="P-value",ylab="CVL",main="Wald test")
points(P.Wald,CVL.Wald[which.max(CVL.Wald)],col="red",pch=17,cex=1.5)
points(P.grid,RCVL1.Wald,col="darkgreen",type="l",lty="dashed")
text(mean(P.grid),mean(RCVL1.Wald),"RCVL1",col="darkgreen")
points(P.grid,RCVL2.Wald,col="blue",type="l",lty="dotted")
text(mean(P.grid),mean(RCVL2.Wald),"RCVL2",col="blue")
AA=paste("Optimal P-value =",as.character(P.Wald))
legend("center",AA,pch=17,col="red",)

######## The plot of CVL (score tests)  #########
Range.score= range(c(CVL.score,RCVL1.score,RCVL2.score),na.rm=TRUE)
plot(P.grid,CVL.score,type="l",ylim=Range.score,xlab="P-value",ylab="CVL",main="Score test")
points(P.score,CVL.score[which.max(CVL.score)],col="red",pch=17,cex=1.5)
points(P.grid,RCVL1.score,col="darkgreen",type="l",lty="dashed")
text(mean(P.grid),mean(RCVL1.score),"RCVL1",col="darkgreen")
points(P.grid,RCVL2.score,col="blue",type="l",lty="dotted")
text(mean(P.grid),mean(RCVL2.score),"RCVL2",col="blue")
AA=paste("Optimal P-value =",as.character(P.grid[which.max(CVL.score)]))
legend("center", AA,pch=17,col="red")

##########################################
#   Predictor construction after feature selection   #
##########################################

######## Feature selection ######
Wald=uni.selection(t.vec,d.vec,X.mat,K=K, P.value=P.Wald,score=FALSE,permutation=TRUE)
score=uni.selection(t.vec,d.vec,X.mat,K=K, P.value=P.score,score=TRUE,permutation=TRUE)
Wald
score
```

```
######## Copula model #######
Wald.copula=dependCox.reg.CV(t.vec,d.vec,X.mat[,names(Wald$beta)],K=5)
c.Wald=round(Wald.copula$c_index,2)
AA=paste("c-index=",as.character(c.Wald))
legend("top",AA,pch=17,col="red")

score.copula=dependCox.reg.CV(t.vec,d.vec,X.mat[,names(score$beta)],K=5)
c.score=round(score.copula$c_index,2)
AA=paste("c-index=",as.character(c.score))
legend("top",AA,pch=17,col="red")

table(X.mat[,"ANXA5"])  ## frequency table for the 63 samples ##

###### Shrinkage estimation ######
Wald.CS=compound.reg(t.vec,d.vec,X.mat[,names(Wald$beta)],K=5)
score.CS=compound.reg(t.vec,d.vec,X.mat[,names(score$beta)],K=5)
Wald.CS
score.CS

#########################################################
##    Predicting Poor or Good Survival for 62 testing samples    ##
#########################################################

### 62 testing samples ###
t.test=Lung$t.vec[!train]
d.test=Lung$d.vec[!train]
X.test=as.matrix(Lung[,-c(1,2,3)][!train,])

par(mfrow=c(3,2))
####### Prediction by Wald #######
X.Wald=X.test[,names(Wald$beta)]
eta.Wald=as.vector(X.Wald%*%Wald$beta)
class.Wald=eta.Wald>median(eta.Wald)
LR.Wald=survdiff(Surv(t.test,d.test) ~ class.Wald)
P.Wald=1-pchisq(LR.Wald$chisq,df=1)
c.Wald=survConcordance(  Surv(t.test,d.test)~eta.Wald  )$concordance

plot(  survfit(Surv(t.test[!class.Wald],d.test[!class.Wald])~1,conf.type="none"),
     ylim=c(0.5,1),col="blue",lwd=2,main="Optimal Wald test",mark.time = TRUE,
     xlab="Months",ylab="Survival probability")
lines(  survfit(Surv(t.test[class.Wald],d.test[class.Wald])~1,conf.type="none"),
      col="red",lwd=2,mark.time = TRUE )
text(35,0.95,paste("P-value =",as.character(round(P.Wald,3))))
text(35,0.88,paste("c-index =",as.character(round(c.Wald,3))))
text(40,0.55,"Poor prognosis",col="red")
```

```
text(40,0.75,"Good prognosis",col="blue")

####### Prediction by score #######
X.score=X.test[,names(score$beta)]
eta.score=as.vector(X.score%*%score$beta)
class.score=eta.score>median(eta.score)
LR.score=survdiff(Surv(t.test,d.test) ~ class.score)
P.score=1-pchisq(LR.score$chisq,df=1)
c.score=survConcordance(  Surv(t.test,d.test)~eta.score  )$concordance

plot(  survfit(Surv(t.test[!class.score],d.test[!class.score])~1,conf.type="none"),
      ylim=c(0.5,1),col="blue",lwd=2,main="Optimal score test",mark.time = TRUE,
      xlab="Months",ylab="Survival probability")
lines(  survfit(Surv(t.test[class.score],d.test[class.score])~1,conf.type="none"),
       col="red",lwd=2,mark.time = TRUE )
text(35,0.95,paste("P-value =",as.character(round(P.score,3))))
text(35,0.88,paste("c-index =",as.character(round(c.score,3))))
text(40,0.55,"Poor prognosis",col="red")
text(40,0.75,"Good prognosis",col="blue")

####### Prediction by Copula + optimal Wald #######
eta.Wald.copula=as.vector(X.Wald%*%Wald.copula$beta)
class.Wald.copula=eta.Wald.copula>median(eta.Wald.copula)
LR.Wald.copula=survdiff(Surv(t.test,d.test) ~ class.Wald.copula)
P.Wald.copula=1-pchisq(LR.Wald.copula$chisq,df=1)
c.Wald.copula=survConcordance(  Surv(t.test,d.test)~eta.Wald.copula  )$concordance

plot(survfit(Surv(t.test[!class.Wald.copula],d.test[!class.Wald.copula])~1,conf.type="none"),
    ylim=c(0.5,1),col="blue",lwd=2,main="Copula + optimal Wald test",mark.time = TRUE,
    xlab="Months",ylab="Survival probability")
lines(survfit(Surv(t.test[class.Wald.copula],d.test[class.Wald.copula])~1,conf.type="none"),
     col="red",lwd=2,mark.time = TRUE)
text(35,0.95,paste("P-value =",as.character(round(P.Wald.copula,3))))
text(35,0.88,paste("c-index =",as.character(round(c.Wald.copula,3))))
text(40,0.52,"Poor prognosis",col="red")
text(40,0.78,"Good prognosis",col="blue")

####### Prediction by Copula + optimal score #######
eta.score.copula=as.vector(X.score%*%score.copula$beta)
class.score.copula=eta.score.copula>median(eta.score.copula)
LR.score.copula=survdiff(Surv(t.test,d.test) ~ class.score.copula)
P.score.copula=1-pchisq(LR.score.copula$chisq,df=1)
c.score.copula=survConcordance(  Surv(t.test,d.test)~eta.score.copula  )$concordance

plot(survfit(Surv(t.test[!class.score.copula],d.test[!class.score.copula])~1,conf.type="none"),
```

```
    ylim=c(0.5,1),col="blue",lwd=2,main="Copula + optimal score test",mark.time = TRUE,
    xlab="Months",ylab="Survival probability")
lines(survfit(Surv(t.test[class.score.copula],d.test[class.score.copula])~1,conf.type="none"),
     col="red",lwd=2,mark.time = TRUE)
text(35,0.95,paste("P-value =",as.character(round(P.score.copula,3))))
text(35,0.88,paste("c-index =",as.character(round(c.score.copula,3))))
text(40,0.57,"Poor prognosis",col="red")
text(40,0.80,"Good prognosis",col="blue")

####### Prediction by CS + optimal Wald #######
eta.Wald.CS=as.vector(X.Wald%*%Wald.CS$beta)
class.Wald.CS=eta.Wald.CS>median(eta.Wald.CS)
LR.Wald.CS=survdiff(Surv(t.test,d.test) ~ class.Wald.CS)
P.Wald.CS=1-pchisq(LR.Wald.CS$chisq,df=1)
c.Wald.CS=survConcordance(  Surv(t.test,d.test)~eta.Wald.CS  )$concordance

plot(survfit(Surv(t.test[!class.Wald.CS],d.test[!class.Wald.CS])~1,conf.type="none"),
    ylim=c(0.5,1),col="blue",lwd=2,main="Shrinkage + optimal Wald test",mark.time = TRUE,
    xlab="Months",ylab="Survival probability")
lines(survfit(Surv(t.test[class.Wald.CS],d.test[class.Wald.CS])~1,conf.type="none"),
     col="red",lwd=2,mark.time = TRUE)
text(35,0.95,paste("P-value =",as.character(round(P.Wald.CS,3))))
text(35,0.88,paste("c-index =",as.character(round(c.Wald.CS,3))))
text(40,0.52,"Poor prognosis",col="red")
text(40,0.75,"Good prognosis",col="blue")

####### Prediction by CS + optimal score #######
eta.score.CS=as.vector(X.score%*%score.CS$beta)
class.score.CS=eta.score.CS>median(eta.score.CS)
LR.score.CS=survdiff(Surv(t.test,d.test) ~ class.score.CS)
P.score.CS=1-pchisq(LR.score.CS$chisq,df=1)
c.score.CS=survConcordance(  Surv(t.test,d.test)~eta.score.CS  )$concordance

plot(survfit(Surv(t.test[!class.score.CS],d.test[!class.score.CS])~1,conf.type="none"),
    ylim=c(0.5,1),col="blue",lwd=2,main="Shrinkage + optimal score test",mark.time = TRUE,
    xlab="Months",ylab="Survival probability")
lines(survfit(Surv(t.test[class.score.CS],d.test[class.score.CS])~1,conf.type="none"),
     col="red",lwd=2,mark.time = TRUE)
text(35,0.95,paste("P-value =",as.character(round(P.score.CS,3))))
text(35,0.88,paste("c-index =",as.character(round(c.score.CS,3))))
text(40,0.52,"Poor prognosis",col="red")
text(40,0.75,"Good prognosis",col="blue")
```

S2. The CVL curve for simulated data (with the R codes):



The codes use simulated data to demonstrate the CVL plot shown above.

```
library(compound.Cox)

n=100
p=500
beta_true=c(rep(0.1,25),rep(-0.1,25),rep(0,p-50))
q0=sum(beta_true==0) ### the number of zero coefficients ###
q1=sum(beta_true>0) ###  the number of positive coefficients ###
q2=sum(beta_true<0) ###  the number of negative coefficients ###

CEN_Bound=1 #=cen~59%#
t.vec=d.vec=numeric(n)
X.mat=X.pathway(n,p,q1,q2)
colnames(X.mat)=c(1:p)

set.seed(10)
for(i in 1:n){
  eta=X.mat[i,]%*%beta_true
  T=rexp(1,rate=exp(eta))
  C=runif(1,min=0,max=CEN_Bound)
  t.vec[i]=min(T,C)
  d.vec[i]=(T<=C)
```

```
}

#########################################################
#    Draw the CVL plot to find the optimal P-value threshold     #
#########################################################

P.grid=seq(0.001,0.01,length=21) ## P-value threshold ##
K=5 ## five-fold cross-validation ##

CVL.score=CVL.Wald=NULL
RCVL1.score=RCVL1.Wald=NULL
RCVL2.score=RCVL2.Wald=NULL

set.seed(10)
for(P in P.grid){
 res.score=uni.selection(t.vec,d.vec,X.mat,K=K,P.value=P,score=TRUE)
 res.Wald=uni.selection(t.vec,d.vec,X.mat,K=K,P.value=P,score=FALSE)
 CVL.score=c(CVL.score,res.score$CVL[1])
 CVL.Wald=c(CVL.Wald,res.Wald$CVL[1])
 RCVL1.score=c(RCVL1.score,res.score$CVL[2])
 RCVL1.Wald=c(RCVL1.Wald,res.Wald$CVL[2])
 RCVL2.score=c(RCVL2.score,res.score$CVL[3])
 RCVL2.Wald=c(RCVL2.Wald,res.Wald$CVL[3])
}
P.Wald=P.grid[which.max(CVL.Wald)]
P.score=P.grid[which.max(CVL.score)]

######## The plot of CVL (Wald tests)  #########
Range.Wald= range(c(CVL.Wald,RCVL1.Wald,RCVL2.Wald),na.rm=TRUE)
par(mfrow=c(1,2))
plot(P.grid,CVL.Wald,type="l",ylim=Range.Wald,xlab="P-value",ylab="CVL",main="Wald test")
points(P.Wald,CVL.Wald[which.max(CVL.Wald)],col="red",pch=17,cex=1.5)
points(P.grid,RCVL1.Wald,col="darkgreen",type="l",lty="dashed")
text(mean(P.grid),mean(RCVL1.Wald),"RCVL1",col="darkgreen")
points(P.grid,RCVL2.Wald,col="blue",type="l",lty="dotted")
text(mean(P.grid),mean(RCVL2.Wald),"RCVL2",col="blue")
AA=paste("Optimal P-value =",as.character(P.Wald))
legend("center",AA,pch=17,col="red",)

######## The plot of CVL (score tests)  #########
Range.score= range(c(CVL.score,RCVL1.score,RCVL2.score),na.rm=TRUE)
plot(P.grid,CVL.score,type="l",ylim=Range.score,xlab="P-value",ylab="CVL",main="Score test")
points(P.score,CVL.score[which.max(CVL.score)],col="red",pch=17,cex=1.5)
points(P.grid,RCVL1.score,col="darkgreen",type="l",lty="dashed")
text(mean(P.grid),mean(RCVL1.score),"RCVL1",col="darkgreen")
```

```
points(P.grid,RCVL2.score,col="blue",type="l",lty="dotted")
text(mean(P.grid),mean(RCVL2.score),"RCVL2",col="blue")
AA=paste("Optimal P-value =",as.character(P.grid[which.max(CVL.score)]))
legend("center", AA,pch=17,col="red")

######## Feature selection at optimal threshold ######
uni.selection(t.vec,d.vec,X.mat,K=K, P.value=P.Wald,score=FALSE,permutation = TRUE)
uni.selection(t.vec,d.vec,X.mat,K=K, P.value=P.score,score=TRUE,permutation = TRUE)
```

S3. R codes for simulation studies (for feature selection):

```
library(compound.Cox)

p=5000 ### the number of genes
beta_true=c(rep(0.1,25),rep(-0.1,25),rep(0,p-50))
#beta_true=c(rep(0.2,25),rep(-0.2,25),rep(0,p-50))
q0=sum(beta_true==0) ### the number of zero coefficients ###
q1=sum(beta_true>0) ###  the number of positive coefficients ###
q2=sum(beta_true<0) ###  the number of negative coefficients ###

simu=function(R,n,K,P.value){

 q_vec=True=False=CVL=CEN_per=numeric(R)
 CEN_Bound=1 #=cen~59%#

 for(r in 1:R){
  t.vec=d.vec=numeric(n)
  set.seed(r)
  X.mat=X.pathway(n,p,q1,q2)
  colnames(X.mat)=c(1:p)
  for(i in 1:n){
   eta=X.mat[i,]%*%beta_true
   T=rexp(1,rate=exp(eta))
   C=runif(1,min=0,max=CEN_Bound)
   t.vec[i]=min(T,C)
   d.vec[i]=(T<=C)
  }
  CEN_per[r]=1-mean(d.vec)
  res=uni.selection(t.vec, d.vec, X.mat, P.value=P.value,K=K,score=TRUE)
  q_vec[r]=length(res$beta)
  CVL[r]=res$CVL[1]
  True[r]= sum( as.numeric(names(res$P))<=q1+q2 )
  False[r]= sum( as.numeric(names(res$P))>q1+q2 )
 }

 c(P=P.value,CEN_Percent=round(mean(CEN_per),2),CVL=mean(CVL),
  q=mean(q_vec),True=mean(True),False=mean(False))

}
```

```
R=50 ### the number of simulation runs
K=5 ### the number of folds in cross-validation
n=100 ### sample size

res=rbind(
  simu(R,n,K,P.value=0.000075),
  simu(R,n,K,P.value=0.0001),
  simu(R,n,K,P.value=0.00025),
  simu(R,n,K,P.value=0.0005),
  simu(R,n,K,P.value=0.00075),
  simu(R,n,K,P.value=0.001),
  simu(R,n,K,P.value=0.0025),
  simu(R,n,K,P.value=0.005),
  simu(R,n,K,P.value=0.0075),
  simu(R,n,K,P.value=0.01),
  simu(R,n,K,P.value=0.025),
  simu(R,n,K,P.value=0.05),
  simu(R,n,K,P.value=0.075)
)

P.grid=res[,"P"]
CVL.grid=res[,"CVL"]
true.grid=res[,"True"]
false.grid=res[,"False"]
res

par(mfrow=c(1,2))
plot(log(P.grid),CVL.grid,type="b",xlab="log(P-value)",ylab="CVL",main="n=100")
temp.CVL=which.max(CVL.grid)
points(log(P.grid[temp.CVL]),CVL.grid[temp.CVL],col="red",pch=17,cex=1.5)
true.CVL=as.character(round(true.grid[temp.CVL],0))
false.CVL=as.character(round(false.grid[temp.CVL],0))
legend("bottom",legend=paste("True genes =",true.CVL,";","False genes =",false.CVL),col="red")
Sen=round(true.grid[temp.CVL]/50,2)
AA=paste("Sensitivity=",true.CVL,"/",as.character(50),"=",as.character(Sen))
Spe=round(true.grid[temp.CVL]/(p-50),2)
AA=paste("Sensitivity=",true.CVL,"/",as.character(50),"=",as.character(Sen))
legend(log(P.grid[temp.CVL])-2,CVL.grid[temp.CVL]-30,AA,pch=17,col="red")


n=200 ### sample size
res=rbind(
  simu(R,n,K,P.value=0.000075),
  simu(R,n,K,P.value=0.0001),
  simu(R,n,K,P.value=0.00025),
  simu(R,n,K,P.value=0.0005),
  simu(R,n,K,P.value=0.00075),
  simu(R,n,K,P.value=0.001),
  simu(R,n,K,P.value=0.0025),
  simu(R,n,K,P.value=0.005),
```

```
  simu(R,n,K,P.value=0.0075),
  simu(R,n,K,P.value=0.01),
  simu(R,n,K,P.value=0.025),
  simu(R,n,K,P.value=0.05),
  simu(R,n,K,P.value=0.075)
)

P.grid=res[,"P"]
CVL.grid=res[,"CVL"]
true.grid=res[,"True"]
false.grid=res[,"False"]
res

plot(log(P.grid),CVL.grid,type="b",xlab="log(P-value)",ylab="CVL",main="n=200")
temp.CVL=which.max(CVL.grid)
points(log(P.grid[temp.CVL]),CVL.grid[temp.CVL],col="red",pch=17,cex=1.5)
true.CVL=as.character(round(true.grid[temp.CVL],0))
false.CVL=as.character(round(false.grid[temp.CVL],0))
legend("bottom",legend=paste("True genes =",true.CVL,";","False genes =",false.CVL),col="red")
Sen=round(true.grid[temp.CVL]/50,2)
AA=paste("Sensitivity=",true.CVL,"/",as.character(50),"=",as.character(Sen))
legend(log(P.grid[temp.CVL]),CVL.grid[temp.CVL]-30,AA,pch=17,col="red")
```

S4. R codes for simulation studies (for prediction):

```
install.packages("compound.Cox")
library(compound.Cox)
library(penalized)

simu=function(R,n,K,P.value){

  p=5000 ### the number of genes
  # p=200 ## for quick test
  beta_true=c(rep(0.1,25),rep(-0.1,25),rep(0,p-50))
  q0=sum(beta_true==0) ### the number of zero coefficients ###
  q1=sum(beta_true>0) ###  the number of positive coefficients ###
  q2=sum(beta_true<0) ###  the number of negative coefficients ###

  q_vec=True=False=CVL=CEN_per=numeric(R)
  CC.c=CS.c=R.c=L.c=numeric(R)
  CC.LR=CS.LR=R.LR=L.LR=numeric(R)
  CEN_Bound=1 #=cen~59%#

  for(r in 1:R){
    t.vec=d.vec=numeric(n)
    set.seed(r)
    X.mat=X.pathway(n,p,q1,q2)
    colnames(X.mat)=c(1:p)
```

```
for(i in 1:n){
  eta=X.mat[i,]%*%beta_true
  T=rexp(1,rate=exp(eta))
  C=runif(1,min=0,max=CEN_Bound)
  t.vec[i]=min(T,C)
  d.vec[i]=(T<=C)
}
CEN_per[r]=1-mean(d.vec)
res=uni.selection(t.vec, d.vec, X.mat, P.value=P.value,K=K,score=TRUE)
q_vec[r]=length(res$beta)
CVL[r]=res$CVL[1]
True[r]= sum( as.numeric(names(res$P))<=q1+q2 )
False[r]= sum( as.numeric(names(res$P))>q1+q2 )
temp=as.numeric(names(res$beta))

### Ridge ###
res_R=optL2(Surv(t.vec,d.vec),penalized=X.mat[,temp],fold=5,trace=FALSE)
beta_R=attributes(res_R$fullfit)$penalized

### Lasso ###
res_L=optL1(Surv(t.vec,d.vec),penalized=X.mat[,temp],fold=5,trace=FALSE)
beta_L=attributes(res_L$fullfit)$penalized

### Compound shrinkage ###
res_CS=compound.reg(t.vec,d.vec,X.mat[,temp],delta_a = 0.1,randomize=FALSE)
beta_CS=res_CS[[2]]

t.test=d.test=numeric(n)
set.seed(r+R)
X.test=X.pathway(n,p,q1,q2)
colnames(X.test)=c(1:p)
for(i in 1:n){
  eta=X.test[i,]%*%beta_true
  T=rexp(1,rate=exp(eta))
  C=runif(1,min=0,max=CEN_Bound)
  t.test[i]=min(T,C)
  d.test[i]=(T<=C)
}

CC.test=X.test[,temp]%*%res$beta
CC.c[r]=survConcordance(  Surv(t.test,d.test)~CC.test  )$concordance
t.o=t.test[order(CC.test)]
d.o=d.test[order(CC.test)]
CC.LR[r]=sqrt( survdiff(Surv(t.o,d.o) ~ c(rep(1,n/2),rep(0,n/2)))$chisq )

CS.test=X.test[,temp]%*%beta_CS
CS.c[r]=survConcordance(  Surv(t.test,d.test)~CS.test  )$concordance
t.o=t.test[order(CS.test)]
d.o=d.test[order(CS.test)]
CS.LR[r]=sqrt( survdiff(Surv(t.o,d.o) ~ c(rep(1,n/2),rep(0,n/2)))$chisq )
```

```
    R.test=X.test[,temp]%*%beta_R
    R.c[r]=survConcordance(  Surv(t.test,d.test)~R.test  )$concordance
    t.o=t.test[order(R.test)]
    d.o=d.test[order(R.test)]
    R.LR[r]=sqrt( survdiff(Surv(t.o,d.o) ~ c(rep(1,n/2),rep(0,n/2)))$chisq )

    L.test=X.test[,temp]%*%beta_L
    L.c[r]=survConcordance(  Surv(t.test,d.test)~L.test  )$concordance
    t.o=t.test[order(L.test)]
    d.o=d.test[order(L.test)]
    L.LR[r]=sqrt( survdiff(Surv(t.o,d.o) ~ c(rep(1,n/2),rep(0,n/2)))$chisq )
   }

   res.train=c(P=P.value,CEN_Percent=round(mean(CEN_per),2),CVL=mean(CVL),
           q=mean(q_vec),True=mean(True),False=mean(False))

   c.index=c(CC=mean(CC.c),Ridge=mean(R.c),Lasso=mean(L.c),CS=mean(CS.c))
   LR.test=c(CC=mean(CC.LR),Ridge=mean(R.LR),Lasso=mean(L.LR),CS=mean(CS.LR))
   list(res.train=res.train,c.index=c.index,LR.test=LR.test)
}

n=100 ### sample size
R=50 ### the number of simulation runs
K=5 ### the number of folds in CV

P.grid=c(0.000075,0.0001,0.00025,0.0005,0.00075,
       0.001,0.0025,0.005,0.0075,0.01,0.025,0.05,0.075)

S1=simu(R,n,K,P.value=0.000075)
S2=simu(R,n,K,P.value=0.0001)
S3=simu(R,n,K,P.value=0.00025)
S4=simu(R,n,K,P.value=0.0005)
S5=simu(R,n,K,P.value=0.00075)
S6=simu(R,n,K,P.value=0.001)
S7=simu(R,n,K,P.value=0.0025)
S8=simu(R,n,K,P.value=0.005)
S9=simu(R,n,K,P.value=0.0075)
S10=simu(R,n,K,P.value=0.01)
S11=simu(R,n,K,P.value=0.025)
S12=simu(R,n,K,P.value=0.05)
S13=simu(R,n,K,P.value=0.075)

S.c=cbind(S1$c.index,S2$c.index,S3$c.index,S4$c.index,S5$c.index,
       S6$c.index,S7$c.index,S8$c.index,S9$c.index,S10$c.index,
       S11$c.index,S12$c.index,S13$c.index)
S.LR=cbind(S1$LR.test,S2$LR.test,S3$LR.test,S4$LR.test,S5$LR.test,
        S6$LR.test,S7$LR.test,S8$LR.test,S9$LR.test,S10$LR.test,
        S11$LR.test,S12$LR.test,S13$LR.test)

CC.c.grid=S.c["CC",]
R.c.grid=S.c["Ridge",]
```

```
CS.c.grid=S.c["CS",]
L.c.grid=S.c["Lasso",]

CC.LR.grid=S.LR["CC",]
R.LR.grid=S.LR["Ridge",]
CS.LR.grid=S.LR["CS",]
L.LR.grid=S.LR["Lasso",]

c.max=max(c(CC.c.grid,R.c.grid,L.c.grid,CS.c.grid))
c.min=min(c(CC.c.grid,R.c.grid,L.c.grid,CS.c.grid))
LR.max=max(c(CC.LR.grid,R.LR.grid,L.LR.grid,CS.LR.grid))
LR.min=min(c(CC.LR.grid,R.LR.grid,L.LR.grid,CS.LR.grid))

par(mfrow=c(1,2))
plot(log(P.grid),CC.LR.grid,ylim=c(LR.min,LR.max),type="b",lty="solid",
   xlab="log(P-value)",ylab="|Z| of the Log-rank test",pch=8,lwd=2)
points(log(P.grid),R.LR.grid,type="b",col="red",pch=17,lwd=2)
points(log(P.grid),CS.LR.grid,type="b",col="blue",pch=16,lwd=2)
points(log(P.grid),L.LR.grid,type="b",col="orange",pch=15,lwd=2)

AA=c("CS","Ridge","CC","Lasso")
BB=c("dashed","dotted","solid","dotdash")
CC=c("blue","red","black","orange")
legend("bottom",AA,lwd=c(2,2),merge = TRUE,col=CC,pch=c(16,17,8,15))

plot(log(P.grid),CC.c.grid,ylim=c(c.min,c.max),type="b",lty="solid",
   xlab="log(P-value)",ylab="c-index",pch=8,lwd=2)
points(log(P.grid),R.c.grid,type="b",col="red",pch=17,lwd=2)
points(log(P.grid),CS.c.grid,type="b",col="blue",pch=16,lwd=2)
points(log(P.grid),L.c.grid,type="b",col="orange",pch=15,lwd=2)

AA=c("CS","Ridge","CC","Lasso")
BB=c("dashed","dotted","solid","dotdash")
CC=c("blue","red","black","orange")
legend("bottom",AA,lwd=c(2,2),merge = TRUE,col=CC,pch=c(16,17,8,15))
```