hdnom: Building Nomograms for Penalized Cox Models with High-Dimensional Survival Data

Supplementary Information

Nan Xiao[1], Qing-Song Xu[1], Miao-Zhu Li[2]

[1] *School of Mathematics and Statistics, Central South University, Changsha, China.*
[2] *Center for Population Health and Aging, Duke University, Durham, NC, United States.*

Contents

1 Penalized Estimation Methods for Cox Models

1.1 From Ridge Regression to Lasso

In real-world high-dimensional regression and classification tasks, the number of variables p in the dataset could be large while the number of samples n is relatively small. In extreme cases, we have $p \gg n$. Such ill-posed problems are challenging for estimators developed under the traditional framework for linear models.

To solve such problems, certain constraints are often imposed on the parameters to be estimated in the statistical model, to reduce the model complexity. In many cases, such constraints yield the effect of variable selection (e.g. some of the estimated coefficients in the linear model will be exactly 0).

Consider the linear regression model (Gaussian case):

$$\mathbf{y} = \mathbf{X}\beta + \varepsilon. \tag{1}$$

Here we assume that $\mathbf{X}_{n\times p}$ is the predictor matrix, and $\mathbf{y}_{n\times 1}$ is the response vector. The parameter of the linear model is $\beta_{p\times 1}$, and $\varepsilon_{n\times 1}$ is the error term.

One intuitive idea for constraining the estimated parameters is by introducing a penalty term in the model based on the magnitudes of the regression coefficients, such that the "less relevant" variables could have a small coefficient. One of the most successful designs of such penalty terms is from the ridge regression [1]:

$$\hat{\beta}(\text{ridge}) = \arg\min_{\beta} \left(\|\mathbf{y} - \mathbf{X}\beta\|_2^2/n + \lambda\|\beta\|_2 \right). \tag{2}$$

The above norm notation $\|\beta\|_s$ indicates $\sum_{i=1}^{p} |\beta_i|^s$.

The ridge estimator will effectively "shrink" the coefficients towards 0, though not exactly 0. The (positive) tuning parameter λ controls the amount of shrinkage. Larger λ means more intensive shrinkage, and vice versa.

For high-dimensional linear models, the ridge estimator (ℓ_2 penalization) usually gives informative results about effect size. However, ridge estimation does not select variables explicitly.

By simply changing the ℓ_2 penalty to ℓ_1 penalty, we have the lasso estimator:

$$\hat{\beta}(\text{lasso}) = \arg\min_{\beta} \left(\|\mathbf{y} - \mathbf{X}\beta\|_2^2/n + \lambda\|\beta\|_1 \right) \tag{3}$$

The lasso regression achieves simultaneous variable selection and effect size estimation, which means the variables not being selected have the coefficient 0. Such property makes the method very attractive, since it sufficiently reduced the model complexity, improves the model interpretability, while still gives a relatively good predictive performance. The ℓ_1 penalty then became very popular and was extensively applied in many more statistical learning problems under more general settings. For a recent review of lasso-inspired methods and related applications, please refer to [2].

The lasso regression corresponds to a unique convex optimization problem. As a milestone in the development of lasso, the Least Angle Regression (LARS) [3] gave a very elegant geometrical explanation and an efficient solution for the lasso.

Even more efficient numerical optimization methods were proposed for solving large-scale lasso-type penalized estimation problems later, including the Coordinate Descent algorithm implemented in the R package `glmnet` [4], and the more recent Alternating Direction Method of Multipliers (ADMM) algorithm [5]. The `hdnom` package is partially depended on the `glmnet` package, which means the shrinkage parameter λ is always automatically tuned over a grid.

1.2 From Lasso to Cox Lasso

The Cox proportional hazards model is:

$$h(t) = h_0(t) \exp(\mathbf{X}\beta) \tag{4}$$

where $h(t)$ is the hazard at time t, $h_0(t)$ is the baseline hazard function shared by all samples. A solution for the Cox model could be obtained by maximizing the partial likelihood for the observed data:

$$\hat{\beta}(\text{Cox}) = \arg\max_{\beta} L(\beta) = \arg\max_{\beta} \prod_{i \in E} \frac{\exp(\mathbf{X}_{j(i)}\beta)}{\sum_{j \in R_i} \exp(\mathbf{X}_j \beta)} \tag{5}$$

where E is the event (e.g. death) set, R_i represents the set of individuals at risk at time point $t_i - 0$. To make the optimization problem easier, the log likelihood is often used instead of the original likelihood.

By constraining the sum of magnitude of β, we have the Cox lasso estimator via maximizing the penalized log partial likelihood [6]:

$$\hat{\beta}(\text{CoxLasso}) = \arg\max_{\beta} \left\{ \frac{2}{n} \left(\sum_{i \in E} \mathbf{X}_{j(i)}\beta - \log \left(\sum_{j \in R_i} \exp(\mathbf{X}_j \beta) \right) \right) - \lambda \|\beta\|_1 \right\}. \tag{6}$$

Note that the penalized likelihood might need some scaling here ($\frac{2}{n}$).

This penalized maximum likelihood estimation for β is aligned in spirit with the penalized linear model (Gaussian case) we investigated above. By changing the penalty term (e.g. $\lambda\|\beta\|_1$), we could easily extend the penalization designed for the Gaussian case to Cox models (although some of the details might need to be slightly modified). Therefore, to simplify the notations, we will only focus on the Gaussian regression case with modifications on the penalty terms in the next sections.

Not too surprisingly, path algorithms, such as LARS-Cox [7], are available for solving the above optimization problem. Besides, coordinate descent algorithms are also applicable here [8].

1.3 From Lasso to Elastic-Net

The lasso is an efficient off-the-shelf method for penalized estimation problems. However, several limitations have been found for lasso estimation, for example:

- Under certain conditions, the lasso procedure is not theoretically consistent;
- The maximum number of selected variables could not be larger than the number of observations (n);
- For correlated variables (which are common in high-dimensional datasets), the selection made by lasso is not stable enough. Sometimes it will select one randomly, instead of including both of the correlated variables.

To overcome such limitations, several important modifications have been made for lasso. For example, the elastic-net penalty [9] was designed as a linear mixture of the ℓ_1 (lasso) and ℓ_2 penalty (ridge):

$$\hat{\beta}(\text{ENet}) = \left(1 + \frac{\lambda_2}{n}\right)\left\{ \arg\min_{\beta} \|\mathbf{y} - \mathbf{X}\beta\|_2^2 + \lambda_2\|\beta\|_2^2 + \lambda_1\|\beta\|_1 \right\} \tag{7}$$

For better interpretability, we could use two parameters, namely λ and α ($0 < \alpha < 1$), as the tuning parameters for controlling the intensity of shrinkage and the sparsity of the model. Essentially, we rewrite the elastic-net penalty as:

$$\lambda\Omega_\alpha(\beta) = \lambda\left(\alpha\|\beta\|_1 + \frac{1}{2}(1-\alpha)\|\beta\|_2^2\right) \tag{8}$$

In the above case, with $\alpha \to 1$, the model tends to be more sparse, with fewer variables selected. In practice, the values of the parameters λ and α are determined by cross-validation with particular evaluation metrics (e.g. RMSE, BIC, etc.).

The major advantages of the elastic-net penalty could be summarized as:

- *Grouping effect* [9]. For highly correlated variables in the dataset, simulations showed that elastic-net tends to select them together as a group, or remove them together as a group, which usually produces a more stable estimation result compared to original lasso.
- A larger number of (more than the number of samples) variables could be selected. This could be potentially beneficial if the sample size n is very small.
- The predictive performance of the model is improved in many cases, partially due to the additionally selected (informative) variables.

The models produced by elastic-net are often less sparse than lasso. There is a possibility that more false positive variables will be included, due to correlations among the variables. However, this could be an inherently difficult problem if there are highly correlated variables in the dataset. Imagine some of them are true positives while some of them are false positives, and from the predictive point of view, there are not many differences between these two types of variables.

1.4 From One-Step Estimation to Adaptive Estimation

Adaptive lasso is another modification for the lasso, and it was introduced as an two-step estimation procedure. The second step is a weighted version of lasso estimation:

$$\hat{\beta}(\text{AdaLasso}) = \arg\min_{\beta} \|\mathbf{y} - \mathbf{X}\beta\|_2^2 + \lambda \sum_{j=1}^{p} \hat{w}_j |\beta_j| \tag{9}$$

Here, $\hat{w}_j$ are the data dependent weights derived from the coefficients β_j' of the first step estimation (ridge estimation are often used). For example, the weights could be derived as $\hat{w}_j = 1/|\beta_j'|^\gamma$, where γ is a tunable parameter.

The adaptive lasso utilizes the information from the first-step estimation to impose more shrinkage for the variables with smaller coefficients, and less shrinkage for the variables with larger coefficients. The modification makes adaptive lasso achieve selection consistency when p is fixed [10]. Such good asymptotic properties of the estimator are formally named as *oracle properties* as was defined in [11].

The adaptive lasso penalty was then further applied in penalized Cox models [12].

Similarly, adaptive elastic-net [13] was proposed as a two-step estimation procedure based on the original elastic-net penalty:

$$\hat{\beta}(\text{AdaEnet}) = \left(1 + \frac{\lambda_2}{n}\right)\left\{ \arg\min_{\beta} \|\mathbf{y} - \mathbf{X}\beta\|_2^2 + \lambda_2\|\beta\|_2^2 + \lambda_1^* \sum_{j=1}^{p} \hat{w}_j |\beta_j| \right\} \tag{10}$$

Compared to one-step estimations (lasso and elastic-net), adaptive lasso and adaptive elastic-net usually have better predictive performance and better false positive control. Ultimately, the idea of multi-step estimation could go even further, with estimators designed with more than two estimation steps [14].

1.5 SCAD, MCP, Snet, and Mnet

The smoothly clipped absolute deviation (SCAD) penalty and minimax concave penalty (MCP) are another two sparsity inducing penalties designed for high-dimensional regression modeling. The SCAD penalty function [11] is defined as

$$\Omega_{\text{SCAD}}(\beta) = \begin{cases} \lambda\beta, & \text{for } \beta \le \lambda \\ \frac{\gamma\lambda\beta - 0.5(\beta^2+\lambda^2)}{\gamma-1}, & \text{for } \lambda < \beta \le \gamma\lambda \\ \frac{\lambda^2(\gamma^2-1)}{2(\gamma-1)}, & \text{for } \beta > \gamma\lambda. \end{cases} \tag{11}$$

with parameters $\lambda \ge 0$ and a scaling factor $\gamma > 2$. The original authors recommended to set γ as 3.7. Similarly, the MCP [15] is defined as

$$\Omega_{\text{MCP}}(\beta) = \begin{cases} \lambda\beta - \frac{\beta^2}{2\gamma}, & \text{for } \beta \le \gamma\lambda \\ \frac{1}{2}\gamma\lambda^2, & \text{for } \beta > \gamma\lambda \end{cases} \tag{12}$$

with parameters $\lambda \geq 0$ and a scaling factor $\gamma > 1$. By computing the first-order derivatives for these two types of penalty functions, we could see the way they work: for variables with very small coefficients, they give (almost) the same amount of penalization as lasso does. As β grows larger, the intensity of the penalization decreases, and finally, there will be no penalization for $\beta > \gamma\lambda$, where β is large enough.

The development of SCAD and MCP provided critical analytical frameworks for evaluating if certain types of similar estimators are good enough in theory, especially the *oracle properties* proposed in [11]. In principle, SCAD and MCP could be both treated as competitive alternatives to the original lasso penalty, and they generally yield less biased estimation results.

In practice, the optimization for SCAD and MCP could be done with coordinate descent algorithms [16], although both of the optimization problems are not exactly convex. Empirically, it takes a longer time to converge than the ℓ_1 and ℓ_2-based methods above.

By combining the ℓ_2 penalty with SCAD and MCP, we will have the Snet [17] and Mnet penalty [18], some theoretical and empirical results have been established for them, though they are less used in practice. In essence, it might be worthy to compare their performance with the results from (adaptive) elastic-net.

1.6 Fused Lasso and Structured Regularization

In some of the high-dimensional datasets we have, the spatial order of the variable is meaningful. For example, the position of the variables in copy number variation (CNV) data [19] reflects the genomic location of the variable, and if a variable at a specific location was selected, we hope that its adjacent variables are also selected.

To incorporate such knowledge into our penalized estimation, an extra constraint could be added to the lasso penalty, and this estimator is named fused lasso [20]:

$$\Omega_{\text{FusedLasso}}(\beta) = \lambda_1 \|\beta\|_1 + \lambda_2 \sum_{i=2}^{p} |\beta_i - \beta_{i-1}|. \tag{13}$$

By further constraining the difference between the coefficients of adjacent variables, fused lasso can usually achieve the specific selection aims we described above. Moreover, such penalties could also be used jointly with other types of penalties mentioned above, such as elastic-net [21].

As a more general formulation, the generalized lasso [22] extended the fused lasso idea to a larger class of problems, by constraining the relationship between variables with a matrix. Other commonly observed structures, such as the hierarchical structure [23], the graph [24] or network structure [25] formed by the variables, has also been successfully incorporated using similar approaches. Such methods could be classified as the *structured regularization* methods, which often reflects our prior knowledge about the explicit or latent structures that existed among the variables.

1.7 Summary

We have described all the penalized estimation methods supported by hdnom for high-dimensional Cox models. In fact, there is no single method consistently working better than other methods in the real world. We should keep in mind that the method which works best with your data, is probably a good choice. Therefore, an empirical model comparison is necessary for choosing a better model. In hdnom, we have the built-in functions `hdnom.compare.validate()` and `hdnom.compare.calibrate()` which supports model comparison via certain validation and calibration procedures.

Additionally, based on our experience on high-dimensional data modeling, we try to offer some basic guidelines and principles on choosing the appropriate method here:

- If you have prior knowledge that the order of the variable in the dataset matters (adjacent variables should be selected together), please try fused lasso first. Less optimal choices could be (adaptive) elastic-net, Snet and Mnet.

- If you do not have any apparent prior knowledge of the correlation structure in the dataset, and you want to build a model with minimal number of selected variables: start with lasso, SCAD, MCP, and see if their results (tAUC, selected variables) are similar or make sense (biologically/clinically). If lasso yielded good results, try adaptive lasso to improve estimation and prediction.

- If you want to further improve the predictive performance of the model and do not care much about introducing more variables into the model, try elastic-net, Snet, Mnet, and tune the sparsity parameter α. If elastic-net gave good results, try adaptive elastic-net and see if it gives even better results (less selected variables, higher tAUC).

Ultimately, a good model should achieve a good balance between the number of selected variables and predictive performance. To be more explicit, for high-dimensional regression modeling, there is usually a trade-off between model sparsity and prediction accuracy. With more variables, the predictive performance of the model could be improved. To have a sparser model (which will have better interpretability), some prediction accuracy often needs to be sacrificed. It is sometimes a difficult task to build a linear model which is very sparse and also have excellent prediction accuracy.

For more specific guidelines on choosing the penalty in Cox regression, please see [26]. A general review of the application of high-dimensional linear models in biology and medicine could be found in [27].

2 Methods for Model Validation and Calibration

Model evaluation, or benchmarking, is a critical part in the predictive modeling process. This step determines the quality of the models and the corresponding nomograms we built [28]. The evaluation should be rigorous, comprehensive, and often involves model validation, model calibration using internal/external datasets.

2.1 Fitting

This method is most frequently used for model calibration in the relevant literature. The method simply uses the entire dataset to fit the Cox model, then make predictions based on this model for the training set. The predicted responses are then compared with the actual observed response to measure the difference.

We believe that this method provides over-optimistic estimates for model performance in most of the cases and could not be qualified as a rigorous method for model evaluation. The reason is, the testing set is identical to the training set, and intuitively, this will usually make the prediction task easier, as if the model could fit the original data well. Therefore, the hdnom package only offers support for model calibration with this method, and we do not really recommend to employ this method to "evaluate" models under any circumstances.

In hdnom, we also support the next three sampling-based methods (bootstrap, cross-validation, and repeated cross-validation). Although they are often used in model validation, they are also made to be applicable to model calibration, too.

2.2 Bootstrap

Bootstrap is a widely-used resampling-based method. Particularly, it is the most used method in previous publications on prognostic model validation.

The procedure is fairly straightforward: sample the same number of observations from the original dataset *with replacement*, use this new dataset as the training set, build the model, make predictions on the *original dataset* (the test set), and compare the differences between the predicted response and the observed response (with metrics like time-dependent AUC). Restart and repeat this procedure for certain times (e.g. 200). After getting a set of tAUC values, we could estimate the empirical mean and variance of them to analyze the model's predictive performance.

Since the bootstrap requires to use the original number of samples to build a model each time and repeat this for a substantial number of times, it will be a slow procedure when the sample size is large. More importantly, sometimes it could give over-optimistic estimates for the predictive performance. Similar to the direct fitting method above, some of the samples in the test set still exist in the training set, although some of them are not selected in the training set by chance each time.

2.3 K-fold Cross-Validation

The k-fold cross-validation (CV) method is popular in the machine learning community. It avoided the problem described above by a training-test splitting design.

The procedure is also simple: split the original dataset randomly into k (k often being 5, 10, or other integers > 3, depending on the sample size) partitions. Use all the $k-1$ partitions as the training set to build a model, predict on the remaining partition (the test set). Move on until k models are built and every partition served as the test set once. Now each sample in the original dataset has a corresponding predicted response, compare them with the observed response using tAUC.

K-fold cross-validation is generally faster than bootstrap since a much less number of models have to be built. It is also more resistant to be over-optimistic since the test set and training set are not intersected at all in each time. However, the number of folds on choice could be controversial sometimes. Also, the reported performance could be unstable, especially if the sample size is small. This is because the dataset splitting may cause the training set and test set to be drastically different distributionally, and this will lead to potential bad predictive performance.

A seemingly good solution for the above problems could be leave-one-out cross-validation (LOOCV). This method only leaves one sample out as the test set each time and use the rest $n-1$ samples as the training set. This is actually an extreme case of the k-fold CV. Unfortunately, it also tends to yield over-optimistic estimates for predictive performance in practice. An intuitive explanation could be, it will be much easier for a good model to make accurate predictions for one sample, with the information from all the other samples available. The one-sample test set is somehow more covered by the training set when being compared to k-fold cross-validation.

2.4 Repeated Cross-Validation

Repeated cross-validation could be a possible remedy for the weaknesses of k-fold CV. It repeats the k-fold CV many times (say, 100), and of course, with randomly different fold splitting schemes every time. This modification makes repeated CV more robust than k-fold CV, and produce stable performance estimates with reduced variance. Repeated cross-validation, like bootstrap, could be slow to run if the repeated number of times is large. Compared to the other methods, we prefer and recommend to use repeated cross-validation in model performance evaluation in practice.

2.5 Further Notes on Cross-Validation

Although k-fold cross-validation and its derivations are routinely used in statistical machine learning research, there have been extensive studies on searching for an even better model performance evaluation approach (with appropriate bias and variance trade-off). For linear models and particularly linear models, one of the most striking theoretical results was obtained in [29], which essentially proved that the leave-one-out

cross-validation (LOOCV) could not yield asymptotically consistent results regarding variable selection. For this case, to achieve consistent selection, a simple yet effective approach is the Monte-Carlo Cross-Validation (MCCV) [30]. This reminds us that it is always important to consider the purpose of the model (e.g. prediction or variable selection) we built when choosing the evaluation scheme. Just like the many penalized estimation methods, there is no single best evaluation approach; however, there could be a comparatively appropriate one for your data and model.

2.6 External Validation and Calibration

To prepare the built survival models ready for real clinical use, model evaluation only within the internal dataset is obviously still not enough. An "independent" external test set (preferably from another data source collected independently) is often required to evaluate the model's performance. In hdnom, the built-in functions `hdnom.external.validate()` and `hdnom.external.calibrate()` are specifically designed for such external evaluation needs (the web application also supports such external evaluation). For practical considerations when performing external validation on prognostic models, please refer to the comprehensive review [31].

2.7 Evaluation Metrics

There are a few types of criteria for prognostic model evaluation. For model validation, we usually measure the model's ability of discrimination, which indicates the differences between the survival curves for different samples or sample groups.

Several measures have been proposed to quantify discrimination. The area under the receiver operating curve (AUC) is considered to be a standard method for such purpose. However, unlike the simple definition of AUC when evaluating classification models, different opinions exist about how to design a better time-dependent AUC estimator for time-to-event data. In hdnom, three types of time-dependent AUC metrics [32, 33, 34] are supported. The selected estimators are mostly based on the idea of cumulative or dynamic control AUC. These estimators usually yield reasonable and stable tAUC estimates in our experiments. For a recent review of the design of such measures and estimators, please refer to [35].

3 Construct Nomograms from Cox Models

In hdnom, the nomogram visualizations are finally created with the support from the `rms` package [36]. However, to correctly map the estimated parameters derived by penalized Cox models to the parallel coordinate system in the nomogram, some intermediate computational steps are still required for all supported types of penalized Cox models. The major components we implemented for this mapping include:

- Survival curve prediction for all supported types of penalized Cox models. This includes computing the predicted survival probabilities and linear predictors for all observations used to fit the Cox model.
- To compute the survival curves, Breslow estimator is used to derive the baseline hazard functions for all supported types of penalized Cox models.
- With the computed survival curves and the linear predictors derived from the penalized Cox model, ordinary least squares is used for creating the mapping between the linear predictors and the selected variables. A similar and reference implementation for survey-weighted Cox models could be found in [37].

4 Implementation Notes

4.1 The R Package

Several high-quality R packages are depended by hdnom to facilitate the implementation of several key features in the package. For building penalized Cox models: the `glmnet` package was used to implement all ℓ_1 and ℓ_2 penalty-based methods; the `penalized` package [38] was used for fitting fused lasso models; the `ncvreg` package [16] was used for the nonconvex penalty (SCAD, MCP, Snet, and Mnet) based models. The `survAUC` package was used for computing the time-dependent AUC metrics for model evaluation. The `rms` package was used for creating the nomograms, and all visualizations for performance evaluation are produced by `ggplot2` [39]. We thank the authors of the above packages for their work.

4.2 Web Application

The web application hdnom.io is built with Shiny [40] and is hosted by shinyapps.io kindly provided by RStudio, Inc. The web application offers an easier way to start for users with less programming experience, and could provide some initial insights when exploring the dataset. We recommend trying the web application with small to medium size datasets since it usually takes a longer time to finish the computation for certain procedures (e.g. nonconvex penalty based models, bootstrap-based model validation). It would be more optimal to perform these analyses using the R package.

References

[1] Arthur E Hoerl and Robert W Kennard. Ridge Regression: Biased Estimation for Nonorthogonal Problems. *Technometrics*, 12(1):55–67, 1970.

[2] Trevor Hastie, Robert Tibshirani, and Martin S. Wainwright. *Statistical Learning with Sparsity: The Lasso and Generalizations*. 2015.

[3] Bradley Efron, Trevor Hastie, Iain Johnstone, Robert Tibshirani, Hemant Ishwaran, Keith Knight, Jean Michel Loubes, Pascal Massart, David Madigan, Greg Ridgeway, Saharon Rosset, J. I. Zhu, Robert A. Stine, Berwin A. Turlach, Sanford Weisberg, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. *The Annals of Statistics*, 32(2):407–499, Apr 2004.

[4] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, 33(1):1–22, 2010.

[5] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011.

[6] Robert Tibshirani. The lasso method for variable selection in the cox model. *Statistics in Medicine*, 16(4):385–395, 1997.

[7] Jiang Gui and Hongzhe Li. Penalized cox regression analysis in the high-dimensional and low-sample size settings, with applications to microarray gene expression data. *Bioinformatics*, 21(13):3001–3008, 2005.

[8] Noah Simon, Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Regularization paths for cox's proportional hazards model via coordinate descent. *Journal of statistical software*, 39(5):1–13, 2011.

[9] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society. Series B: Statistical Methodology*, 67(2):301–320, Apr 2005.

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

[11] Jianqing Fan and Runze Li. Variable selection via nonconcave penalized likeli-

hood and its oracle properties. *Journal of the American statistical Association*, 96(456):1348–1360, 2001.

[12] Hao Helen Zhang and Wenbin Lu. Adaptive lasso for cox's proportional hazards model. *Biometrika*, 94(3):691–703, 2007.

[13] Hui Zou and Hao Helen Zhang. On the adaptive elastic-net with a diverging number of parameters. *The Annals of Statistics*, 37(4):1733, 2009.

[14] Nan Xiao and Qing-Song Xu. Multi-step adaptive elastic-net: reducing false positives in high-dimensional variable selection. *Journal of Statistical Computation and Simulation*, 85(18):3755–3765, 2015.

[15] Cun-Hui Zhang. Nearly unbiased variable selection under minimax concave penalty. *The Annals of Statistics*, 38(2):894–942, 2010.

[16] Patrick Breheny and Jian Huang. Coordinate descent algorithms for nonconvex penalized regression, with applications to biological feature selection. *The Annals of Applied Statistics*, 5(1):232, 2011.

[17] Lingmin Zeng and Jun Xie. Group variable selection via SCAD-ℓ_2. *Statistics*, 48(1):49–66, 2012.

[18] Jian Huang, Patrick Breheny, Shuangge Ma, and Cun-hui Zhang. The Mnet Method for Variable Selection. *Statistica Sinica*, 26:903–923, 2016.

[19] Robert Tibshirani and Pei Wang. Spatial smoothing and hot spot detection for CGH data using the fused lasso. *Biostatistics*, 9(1):18–29, Jan 2008.

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

[21] Guang-Hui Fu and Qing-Song Xu. Grouping Variable Selection by Weight Fused Elastic Net for Multi-Collinear Data. *Communications in Statistics - Simulation and Computation*, 41(2):205–221, 2012.

[22] Ryan J. Tibshirani and Jonathan Taylor. The solution path of the generalized lasso. *The Annals of Statistics*, 39(3):1335–1371, 2011.

[23] S Wang, B Nan, N Zhu, and J Zhu. Hierarchically penalized cox regression with grouped variables. *Biometrika*, 96(2):307–322, 2009.

[24] Caiyan Li and Hongzhe Li. Variable selection and regression analysis for graph-structured covariates with an application to genomics. *The Annals of Applied Statistics*, 4(3):1498, 2010.

[25] Wei Pan, Benhuai Xie, and Xiaotong Shen. Incorporating predictor network in penalized regression with application to microarray data. *Biometrics*, 66(2):474–484, 2010.

[26] Axel Benner, Manuela Zucknick, Thomas Hielscher, Carina Ittrich, and Ulrich Mansmann. High-dimensional cox models: The choice of penalty as part of the model building process. *Biometrical Journal*, 52(1):50–69, Feb 2010.

[27] Peter Bühlmann, Markus Kalisch, and Lukas Meier. High-Dimensional Statistics with a View Toward Applications in Biology. *Annual Review of Statistics and Its Application*, 1(1):255–278, 2014.

[28] Alexia Iasonos, Deborah Schrag, Ganesh V Raj, and Katherine S Panageas. How To Build and Interpret a Nomogram for Cancer Prognosis. *Journal of Clinical Oncology*, 26(8):1364–1370, Mar 2015.

[29] Jun Shao. Linear model selection by cross-validation. *Journal of the Americal Statistical Association*, 88(422):486–494, 1993.

[30] Qing-Song Xu and Yi-Zeng Liang. Monte Carlo cross validation. *Chemometrics and Intelligent Laboratory Systems*, 56(1):1–11, 2001.

[31] Patrick Royston and Douglas G Altman. External validation of a Cox prognostic model: principles and methods. *BMC medical research methodology*, 13(1):33, Dec 2013.

[32] Lloyd E. Chambless and Guoqing Diao. Estimation of time-dependent area under the ROC curve for long-term risk prediction. *Statistics in Medicine*, 25(20):3474–3486, Oct 2006.

[33] Hajime Uno, Tianxi Cai, Lu Tian, and L. J Wei. Evaluating Prediction Rules for t-Year Survivors With Censored Regression Models. *Journal of the American Statistical Association*, 102(478):527–537, 2007.

[34] Xiao Song and Xiao Hua Zhou. A semiparametric approach for the covariate specific ROC curve with survival outcome. *Statistica Sinica*, 18:947–965, 2008.

[35] Matthias Schmid, Hans A. Kestler, and Sergej Potapov. On the validity of time-

dependent AUC estimators. *Briefings in Bioinformatics*, 16(1):153–168, Jan 2013.

[36] Frank E Harrell. *Regression modeling strategies: with applications to linear models, logistic regression, and survival analysis*. Springer Science & Business Media, 2015.

[37] Marinela Capanu and Mithat Gönen. Building a nomogram for survey-weighted cox models using R. *Journal of Statistical Software*, 64:1–17, 2015.

[38] Jelle J Goeman. ℓ_1 penalized estimation in the cox proportional hazards model. *Biometrical Journal*, 52(1):70–84, 2010.

[39] Hadley Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2009.

[40] Winston Chang, Joe Cheng, J Allaire, Yihui Xie, and Jonathan McPherson. Shiny: web application framework for R. *R package version 0.11*, 1, 2015.