Journal of Statistical Software

April 2012, Volume 47, Issue 9. http://www.jstatsoft.org/

Coordinate Descent Methods for the Penalized Semiparametric Additive Hazards Model

Anders Gorst-Rasmussen
Aalborg University

Thomas H. Scheike
University of Copenhagen

Abstract

For survival data with a large number of explanatory variables, lasso penalized Cox regression is a popular regularization strategy. However, a penalized Cox model may not always provide the best fit to data and can be difficult to estimate in high dimension because of its intrinsic nonlinearity. The semiparametric additive hazards model is a flexible alternative which is a natural survival analogue of the standard linear regression model. Building on this analogy, we develop a cyclic coordinate descent algorithm for fitting the lasso and elastic net penalized additive hazards model. The algorithm requires no nonlinear optimization steps and offers excellent performance and stability. An implementation is available in the R package **ahaz**. We demonstrate this implementation in a small timing study and in an application to real data.

Keywords: survival, additive hazards model, lasso, coordinate descent.

1. Introduction

With the increasing interest in high-throughput biomarker research, there is a growing need for simple and efficient statistical methods for relating a survival time endpoint to a large number of explanatory variables. Variable selection methods such as lasso (Tibshirani 1997) or SCAD (Fan and Li 2001) offer convenient means of imposing additional regularity via penalization such that well-known regression models can be straightforwardly adapted to high-dimensional data. By now, many standard survival regression models have been subjected to various penalization strategies (Li 2008), yet the Cox proportional hazards model continues to serve as a reference model and the main target of theoretical, algorithmic, and applied research on penalized survival regression. Although the Cox model is both flexible and simple to interpret, alternative modeling strategies deserve a wider appreciation for a number of reasons. For example, Ma, Huang, Shi, Li, and Shia (2010) recently pointed out a fact

which is well known from a lower-dimensional setting: that a Cox model may not always provide a satisfactory fit to a high-dimensional data set. Moreover, with the increasingly high-dimensional data available today, the intrinsically nonlinear Cox model is a peculiar reference model in terms of the computational efficiency and stability of fitting procedures. A range of algorithms have been developed for fitting penalized Cox models (Gui and Li 2005; Park and Hastie 2007; Sohn, Kim, Jung, and Park 2009; Goeman 2010, and others) but their computational performance is limited by the use of costly Newton-Raphson iterations or similar to deal with the penalized partial likelihood.

Only recently did Simon, Friedman, Hastie, and Tibshirani (2011) describe an impressively fast algorithm for fitting the penalized Cox model which combines iteratively reweighted least squares with cyclic coordinate descent. Cyclic coordinate descent optimizes a convex loss function by solving all coordinatewise optimization problems in an iterative manner. While not a new technique in the context of penalized regression (see the references in Friedman, Hastie, and Tibshirani 2010), cyclic coordinate descent has recently been rediscovered for its ability to efficiently handle even very high-dimensional problems when carefully implemented. For generalized linear models and the Cox model, software for performing coordinate descent-based penalized estimation is available in the package **glmnet** (Friedman *et al.* 2010) for the R system for statistical computing (R Development Core Team 2012).

In this paper, we develop a cyclic coordinate descent algorithm for the elastic net penalized variant of a flexible but less well-known alternative to the Cox model, the so-called semiparametric additive hazards model (Lin and Ying 1994; McKeague and Sasieni 1994). This model asserts a hazard function given by the sum of some baseline hazard function and a regression function of the explanatory variables. It is a survival analogue of the standard linear regression model and leads to natural estimating equations which are surprisingly similar to the normal equations. The flexibility and computational parsimony of the additive hazards model makes it a useful tool on which to base regularization methods for high-dimensional survival data (Ma, Kosorok, and Fine 2006; Leng and Ma 2007; Martinussen and Scheike 2009, 2010). We describe how computational tricks for the penalized linear regression model can be adapted to obtain a very efficient and stable coordinate descent method for fitting the elastic net penalized additive hazards model. In contrast to coordinate descent methods for the penalized Cox model, convergence is theoretically guaranteed for our algorithm. The algorithm has been implemented in C to interface with the R package **ahaz**, available from the Comprehensive R Archive Network at `http://CRAN.R-project.org/package=ahaz`. We provide examples of its usage and performance on simulated and real data.

2. The semiparametric additive hazards model

Suppose that we observe $(T_1, \delta_1, Z_1), \ldots, (T_n, \delta_n, Z_n)$ where T_i is a (right-censored) survival time, δ_i is the indicator which is 1 if subject i experiences an event at time T_i and 0 otherwise, and $Z_i \in \mathbb{R}^p$ is a vector of explanatory variables. To simplify notation, we will describe each pair (T_i, δ_i) via the counting process $N_i(t) := I(T_i \leq t)\delta_i$ and the at-risk-process $Y_i(t) := I(t \leq T_i)$ where I denotes the indicator function. The counting process integral $\int_0^t f(s)\mathrm{d}N_i(s)$ is then simply a notationally convenient way of writing $f(T_i)I(T_i \leq t)\delta_i$.

The semiparametric additive hazards model (Lin and Ying 1994; McKeague and Sasieni 1994)

asserts a conditional hazard function of the form

$$\lambda(t|Z_i) = \lambda_0(t) + Z_i^\top \beta^0;$$

with λ_0 some unspecified baseline hazard constituting the nonparametric part of the model. Lin and Ying (1994) proposed to perform estimation in this model via estimating equations which mimick the score equations for the Cox model. Specifically, they proposed to estimate β^0 as the root of the pseudo-score function

$$U(\beta) := \int_0^\infty \sum_{i=1}^n Z_i\{\mathrm{d}N_i(t) - Y_i(t)\mathrm{d}\hat{\Lambda}_0(t;\beta) - Y_i(t)Z_i^\top \beta \mathrm{d}t\}, \tag{1}$$

where $\hat{\Lambda}_0$ is a Breslow-type estimator of the cumulative baseline hazard $\int_0^t \lambda_0(s)\mathrm{d}s$,

$$\hat{\Lambda}_0(t;\beta) := \int_0^t \frac{\sum_{i=1}^n \{\mathrm{d}N_i(s) - Y_i(s)Z_i^\top \beta \mathrm{d}s\}}{\sum_{i=1}^n Y_i(s)}. \tag{2}$$

Solving $U(\beta) = 0$ is equivalent to solving the $p \times p$ linear system of equations

$$D\beta = d \tag{3}$$

taking

$$D := \int_0^\infty \sum_{i=1}^n \{Z_i - \bar{Z}(t)\}\{Z_i - \bar{Z}(t)\}^\top Y_i(t)\mathrm{d}t \tag{4}$$

$$d := \int_0^\infty \sum_{i=1}^n \{Z_i - \bar{Z}(t)\}\mathrm{d}N_i(t); \tag{5}$$

with $\bar{Z}(t) := \sum_{i=1}^n Z_i Y_i(t) / \sum_{i=1}^n Y_i(t)$ the at-risk-average of the Z_is. The estimator obtained from (3) can be shown to be root-n consistent by martingale arguments (Lin and Ying 1994). The estimating equation in (3) is attractive for several reasons. Not only does it provide an explicitly calculable estimator in a flexible semiparametric model; it is also analytically very similar to the normal equations $(X^\top X)\beta = X^\top y$ for the classical linear regression model $y = X\beta^0 + \varepsilon$. In fact, defining 'responses' $y_i = \mathrm{d}N_i(t)$ and 'explanatory variables' $X_i = (Z_i - \bar{Z}(t))Y_i(t)$, it is seen that (3) is simply a time-averaged version of the normal equations. The similarity between (3) and the normal equations was exploited by Leng and Ma (2007) and Martinussen and Scheike (2009) to construct a lasso penalized estimator for the additive hazards model. They noted that solving (3) is equivalent to minimizing the loss function

$$L(\beta) = \beta^\top D\beta - 2\beta^\top d; \tag{6}$$

leading to a lasso penalized variant with a loss function of the form

$$L_{\mathrm{pen}}(\beta;\lambda) = L(\beta) + \lambda\|\beta\|_1. \tag{7}$$

Here $\|\cdot\|_1$ is the ℓ^1-norm while $\lambda \geq 0$ is a parameter controlling the degree of regularization. Because of geometric properties of the ℓ^1-norm, the lasso penalized estimator $\operatorname{argmin}_\beta L_{\mathrm{pen}}(\beta;\lambda)$ does shrinkage and variable selection simultaneously (Tibshirani 1997). For large values of λ,

most lasso regression coefficients will be exactly zero – as λ grows smaller, the lasso regression coefficients become increasingly similar to their unpenalized counterparts.

Leng and Ma (2007) and Martinussen and Scheike (2009) proposed to use the lasso-LARS algorithm (Efron, Hastie, Johnstone, and Tibshirani 2004) to calculate the lasso penalized estimator $\text{argmin}_\beta L_{\text{pen}}(\beta;\lambda)$. The lasso-LARS algorithm for the standard linear regression model is easily adapted to work with the additive hazards model by supplying pre-computed versions of D (in place of the covariance matrix) and d (in place of the covariate-response inner products). However, pre-computation of D may be unfeasible for large p. Even without pre-computation, the computational cost of lasso-LARS is similar to that of solving the unpenalized regression problem which is substantial for large p. In the following, we propose cyclic coordinate descent as a much more efficient alternative.

3. Model fitting via cyclic coordinate descent

Since the extension is straightforward, we will work with a variant of (7) which includes an ℓ^2 penalty term. That is, we consider the problem of minimizing the following penalized loss:

$$L_{\text{pen}}(\beta;\lambda,\alpha) := L(\beta) + \lambda\alpha\|\beta\|_1 + \frac{1}{2}\lambda(1-\alpha)\|\beta\|_2^2, \quad 0 < \alpha < 1. \tag{8}$$

We denote henceforth

$$\hat{\beta}(\lambda) := \text{argmin}_\beta L_{\text{pen}}(\beta;\lambda,\alpha). \tag{9}$$

The ℓ^1/ℓ^2 penalization in (8) is known as elastic net penalization (Zou and Hastie 2005). When $\alpha = 1$, the loss function reduces to lasso penalized loss. If $\alpha < 1$, the loss function favors joint selection of highly correlated variables. This follows by similar arguments as in Zou and Hastie (2005), utilizing the heuristic interpretation of D as a time-averaged covariance matrix. Choosing $\alpha < 1$ is often more appropriate in terms of stability and interpretation for problems where many of the explanatory variables are highly correlated. We have omitted the dependence of α in the left-hand side of (9) for notational simplicity.

Cyclic coordinate descent is a numerical optimization technique which approximates the minimum of a function $f\colon \mathbb{R}^p \to \mathbb{R}$ by iteratively for $k = 0, 1, 2, \ldots$ cycling through the p coordinatewise optimization problems

$$x_j^{(k)} := \text{argmin}_{x_j} f\big(x_1^{(k)}, \ldots, x_{j-1}^{(k)}, x_j, x_{j+1}^{(k-1)}, \ldots, x_p^{(k-1)}\big), \quad j = 1, 2, \ldots, p; \tag{10}$$

fixing for the update of the j-th coordinate all other coordinates at their most recent value. For a convex f satisfying certain separability conditions, the iterates $x^{(k)}$ converge to $\text{argmin}_{x\in\mathbb{R}^p} f(x)$, irrespective of $x^{(0)}$ (Tseng 1988). It suffices that f is a convex and continuously differentiable function subjected to elastic net penalization.

To use cyclic coordinate descent to calculate (9), simply observe that

$$\frac{\partial L_{\text{pen}}}{\partial \beta_j} = d_j - \sum_{i\neq j} \beta_i D_{ij} + \lambda\alpha\,\text{sign}(\beta_j) + \lambda(1-\alpha)\beta_j.$$

It follows that the updating rule (10), for a given value of (λ,α), takes on the form

$$\beta_j^{(k)} := \frac{\mathcal{S}\Big(d_j - \sum_{i<j}\beta_i^{(k)} D_{ij} + \sum_{i>j}\beta_i^{(k-1)} D_{ij}, \lambda\alpha\Big)}{D_{jj} + \lambda(1-\alpha)}, \qquad j = 1, 2, \ldots, p;$$

where $\mathcal{S}$ denotes the soft-thresholding operator

$$\mathcal{S}(x, y) := \text{sign}(x)(|x| - y)_+.$$

While convexity ensures theoretically that $\beta^{(k)}$ converges to $\hat{\beta}(\lambda)$, convergence can be very slow if $\beta^{(0)}$ is poorly chosen. Fundamental to ensuring rapid convergence and stability of coordinate descent are the following two structural properties of the elastic net problem:

1. If λ is sufficiently large then $\hat{\beta}(\lambda) = 0$ (sparsity).
2. If $\lambda_1 \approx \lambda_2$ then $\hat{\beta}(\lambda_1) \approx \hat{\beta}(\lambda_2)$ (continuity of regularization paths; Efron *et al.* (2004)).

Hence, $\hat{\beta}(\tilde{\lambda})$ for some $\tilde{\lambda}$ can be calculated efficiently and stably via coordinate descent by calculating a pointwise regularization path $\hat{\beta}(\lambda_{\max}), \ldots, \hat{\beta}(\tilde{\lambda})$ at a grid of closely spaced λ-values; starting out with some large $\lambda_{\max}$ so that $\hat{\beta}(\lambda_{\max}) = 0$ and using the most recent solution $\hat{\beta}(\lambda_{l-1})$ as the initial value in the coordinate descent algorithm for $\hat{\beta}(\lambda_l)$. This idea of using 'warm starts' was discussed in more detail by Friedman, Hastie, Höfling, and Tibshirani (2007) and Friedman *et al.* (2010). For the penalized loss (8), it is easily seen that we obtain $\hat{\beta}(\lambda_{\max}) \equiv 0$ by taking

$$\lambda_{\max} := \max_{1 \le j \le p} |d_j|.$$

Following Simon *et al.* (2011), we consider an exponentially decreasing sequence of regularization parameters of length m from $\lambda_{\max}$ to some $\lambda_{\min} < \lambda_{\max}$ such that

$$\lambda_l := \lambda_{\max}\Big(\frac{\lambda_{\min}}{\lambda_{\max}}\Big)^{l/m}, \quad l = 0, \ldots, m-1. \tag{11}$$

If we denote $\varepsilon := \lambda_{\min}/\lambda_{\max}$, a reasonable, although arbitrary, choice is to take $m = 100$ and $\varepsilon = 0.0001$ if $n < p$ and $\varepsilon = 0.05$ if $p \ge n$.

Naively, one would run the coordinate descent algorithm over all p coordinates (i.e., using all p variables) to obtain $\hat{\beta}(\lambda_0), \ldots, \hat{\beta}(\lambda_m)$. This is clearly undesirable for large p since it requires calculation of the entire matrix D. However, given $\hat{\beta}(\lambda)$ for some λ, the Karush-Kuhn-Tucker (KKT) conditions for the constrained optimization problem (8) imply that $\hat{\beta}_j(\lambda) = 0$ iff

$$\Big|d_j - \sum_{i \ne j} \hat{\beta}_i(\lambda) D_{ij}\Big| \le \lambda\alpha. \tag{12}$$

This leads to the active set strategy (Friedman *et al.* 2007): we maintain at all times a set A of 'active variables' which are included in the coordinate descent algorithm, starting out with $A := \emptyset$ at $\lambda_{\max}$. Upon convergence of coordinate descent among variables in A, we check (12) for each variable in $\{1, \ldots, p\} \backslash A$. If there are no violations, we have the final solution. If there are violations, we add the violators to A and restart the coordinate descent algorithm. With this approach, it is seen from (12) that we need only calculate rows $D_{j\bullet}$ for $j \in A$.

A basic coordinate descent algorithm for the additive hazards model is thus the following.
Initialize $A := \emptyset$, $\lambda_{\max} := \max_{1 \le j \le p} |d_j|$, and $\beta^{(0)}(\lambda_{\max}) := 0$.
For $l = 0, \ldots, m-1$ do

1. Set $\lambda_l := \lambda_{\max}\varepsilon^{l/m}$. Do for $k = 0, 1, \ldots,$ until convergence

(a) For $j \in A$, update

$$\beta_j^{(k)}(\lambda_l) := \frac{\mathcal{S}\Big(d_j - \sum_{i\in A, i<j} \beta_i^{(k)}(\lambda_l) D_{ij} - \sum_{i\in A, i>j} \beta_i^{(k-1)}(\lambda_l) D_{ij}, \lambda_l \alpha\Big)}{D_{jj} + \lambda_l(1-\alpha)}. \quad (13)$$

2. Set $\tilde{\beta} := \beta^{(k)}(\lambda_l)$ and for $j \in \{1, \ldots, p\} \backslash A$, calculate

$$V := \Big\{ j \notin A : \Big| d_j - \sum_{i\in A} \tilde{\beta}_i D_{ij} \Big| > \lambda_l \alpha \Big\}.$$

If $V \neq \emptyset$, calculate $D_{j1}, \ldots, D_{jp}$ for $j \in A$, then adjoin V to A and go back to step 1, using $\tilde{\beta}$ as a warm start. Otherwise set $\hat{\beta}(\lambda_l) := \tilde{\beta}$ and $\beta^{(0)}(\lambda_{l+1}) := \tilde{\beta}$, and increment l.

Various stopping criteria can be used in step 1; either based on the relative change in the individual coefficient estimates or based on the relative change in the penalized loss function. We prefer the latter since the loss function is less susceptible to instabilities when many variable are included or when near a saturated fit. Specifically, we declare convergence when the relative change in $L_{\text{pen}}\{\beta^{(k)}(\lambda)\}$ from one value of k to the next is less than 10^{-5}.

Note that for $\alpha = 1$, at most $n-1$ variables can be included in the model, by the nature of the lasso penalized problem. In most cases, the user will specify some maximum number of variables to include which is strictly less than n.

3.1. Efficient calculation of D

The calculation of rows in the matrix D is the primary bottleneck of our basic coordinate descent algorithm for large p. In contrast to the partial likelihood in the Cox model, which essentially only depends on data at failure times, calculation of D uses data at both censoring and failure times. Fortunately, it turns out that (4) can still be evaluated rather efficiently.

Suppose that survival times are ordered such that $T_1 > T_2 > \cdots > T_n$, assuming no ties. Denote $\Delta_k := T_k - T_{k+1}$ (taking $T_{n+1} := 0$) and assume that variables are centered so that $\sum_{i=1}^n Z_i = 0$. By applying the summation by parts formula, we obtain

$$\begin{aligned}
D_{ij} &= \sum_{k=1}^n Z_{jk}\Big(Z_{ik}\int_0^\infty Y_k(t)\mathrm{d}t\Big) - \int_0^\infty \bar{Z}_i(t)\sum_{k=1}^n Z_{jk}Y_k(t)\mathrm{d}t \\
&= \sum_{k=1}^n Z_{jk}(Z_{ik}T_k) + \sum_{k=1}^n \Big(\Delta_k k^{-1}\sum_{h=1}^k Z_{ih}\Big)\Big(\sum_{h=1}^k Z_{jh}\Big) \\
&= \sum_{k=1}^n Z_{jk}(Z_{ik}T_k) + \sum_{k=1}^{n-1}\Big(\sum_{l=1}^k \Delta_l l^{-1}\sum_{m=1}^l Z_{im}\Big) Z_{j,k+1} \\
&= \sum_{k=1}^n Z_{j,k}\tilde{Z}_{ik};
\end{aligned}$$

where we have defined

$$\tilde{Z}_{i1} := Z_{i1}T_1, \qquad \text{and } \tilde{Z}_{ik} := Z_{j,k}T_k + \sum_{l=1}^{k-1}\Delta_l l^{-1}\sum_{m=1}^l Z_{im}, \quad 2 \le k \le n.$$

Hence, if we pre-calculate and store $\tilde{Z}_{i1}, \ldots, \tilde{Z}_{in}$, the subsequent calculation of each matrix element D_{ij} can be accomplished at the modest cost of $2n$ arithmetic operations.

3.2. Increasing efficiency via improved KKT checks

While our basic coordinate descent algorithm is already quite efficient, there is room for improvement. Denote by $\tilde{p}$ the size of the active set A at $\lambda_{\min}$. In retrospect, we need only $\tilde{p}^2$ entries in the matrix D to construct a regularization path; the remaining $(p-\tilde{p})\cdot\tilde{p}$ entries are used only for the KKT checks (12). A substantially more efficient KKT check can be devised by noting from (4) that

$$\sum_{i=1}^{p} D_{ij}\hat{\beta}_i(\lambda) = \sum_{i=1}^{n} Z_{ji} r_i(\lambda); \tag{14}$$

where

$$r_i(\lambda) := \int_0^\infty Y_i(t)\{R_i^\lambda - \bar{R}^\lambda(t)\}\mathrm{d}t, \tag{15}$$

taking $R_i^\lambda := Z_i^\top \hat{\beta}(\lambda)$ to be the linear risk score of the i-th subject. Formulas as in Section 3.1 can be used for evaluating (15). Substituting (14) in (12), it follows that we can perform the necessary KKT checks by calculating the vector $r(\lambda)$ and subsequently evaluating $p-|A|$ inner products between n-vectors. Whenever a new variable j enters the model, symmetry of the matrix D implies that we need only calculate D_{ij} for $i \in A$ to be able to run the coordinate descent updates (13). This is a substantial improvement over the basic coordinate descent algorithm in which the entire row $D_{j\bullet}$ must be calculated for each new variable j.

An issue not addressed by this improved strategy is that KKT checks often fail. In fact, they fail at least whenever a new variable enters the model and in practice much more frequently. A failed check leads to a restart of the coordinate descent loop. Although another run of coordinate descent is rarely very expensive when using warm starts, calculating $p-|A|$ inner products between n-vectors for the next KKT check is costly. The cost could be reduced if we could first run the coordinate descent/check/restart procedure on a set of variables which is larger than the active set but still smaller than p; and outside which KKT violations are rare. Tibshirani, Bien, Friedman, Hastie, Simon, Taylor, and Tibshirani (2012) recently showed how to construct such a set. Adapting their formulas to the present problem, given some $\gamma > \lambda$, they proposed the following sequential strong condition

$$|d_j - Z_j^\top r(\gamma)| \leq \lambda - (\gamma - \lambda) = 2\lambda - \gamma; \tag{16}$$

and argued that if a variable j satisfies this condition then typically $\hat{\beta}_j(\lambda) = 0$. The sequential strong condition is not failsafe and (16) may hold true if $\hat{\beta}_j(\lambda) \neq 0$. The point is that this happens rarely. Consequently, by introducing the strong set

$$S := \{j \notin A : |Z_j^\top r(\gamma)| > 2\lambda - \gamma\} \cup A,$$

we may further improve efficiency of coordinate descent via the following strategy for each λ:

1. Run coordinate descent for variables in A until convergence.
2. Check for violations of KKT conditions among variables in S only, using (14). If violations occur, add violators to A and go back to step 1, using the current solution as a warm start. Otherwise proceed to step 3.

3. Check for violation of KKT conditions among variables in $\{1, \ldots, p\} \backslash S$ using (14). If violations occur, add violators to A, update S, and go to step 1, using the current solution as a warm start. Otherwise we have obtained the solution for this value of λ.

This strategy is an improvement since we tend to restart the algorithm fewer times in step 3. Accordingly, fewer inner products must be calculated. Other approximate discarding rules than (16) could be used instead since we always conclude by running a fail-safe check of KKT conditions among all variables.

3.3. Implementation in ahaz

The optimized version of the algorithm described in this section has been implemented in C to interface with the R package **ahaz** via the wrapper function `ahazpen`. Since all calculations are done in C, the code can easily be adapted to work with other front-ends than R.

The bottleneck of the algorithm is calculating the roughly p inner products between n-vectors. This can account for 50%–90% of the computation time. As also noted by Tibshirani *et al.* (2012), simultaneous inner product evaluations are embarrassingly parallel, suggesting good scalability of the algorithm. We have implemented the inner product evaluations via level 2 calls to the **BLAS** libraries linked to R, thus enabling the user to improve speed of `ahazpen` further by linking R against high-performance **BLAS** libraries such as **GotoBLAS** or **ATLAS**.

4. Additional details

The implementation of cyclic coordinate descent provided in `ahazpen` supports a similar set of options as **glmnet** (Friedman *et al.* 2010); including observation weighting and differential penalization. Specifically, for nonnegative weights $w_1, \ldots, w_p$, `ahazpen` can accommodate a penalized loss function of the form

$$L(\beta) + \lambda \sum_{j=1}^{p} w_j |\beta_j|.$$

In the simplest case, differential penalization can be used to completely exclude a variable from penalization (by setting $w_j := 0$), offering a simple alternative to the more sophisticated approach of unpenalized adjustment discussed by Martinussen and Scheike (2009). Differential penalization also enables implementation of techniques such as adaptive lasso (Zou 2006).

4.1. Delayed entry

An approach which is common in, for example, survival epidemiological studies is adjust for the age of study subjects by using it as a time axis in hazard regression models. This is popularly known as delayed entry (or left-truncation) and requires us to consider data of the form $(S_1, T_1, \delta_1), \ldots, (S_n, T_n, \delta_n)$ where $0 \leq S_i < T_i$ is the entry time of the i-th individual. By keeping $N_i(t) = I(T_i \leq t \wedge \delta_i = 1)$ but setting $Y_i(t) = I(S_i \leq t \leq T_i)$, the regression models described in Section 2 extend straightforwardly to the delayed entry case.

Computer implementation of delayed entry is slightly more involved. Define for $i = 1, 2, \ldots, n$

the following collection of 'pseudo observations':

$$\begin{aligned} Y_i^*(t) &:= I(0 \le t \le T_i), & Y_{i+n}^*(t) &:= -I(0 \le t < S_i); \\ N_i^*(t) &:= N_i(t), & N_{i+n}^*(t) &:= 0; \\ Z_i^* &:= Z_i, & Z_{i+n}^* &:= Z_i. \end{aligned}$$

Since $Y_i(t) = Y_i^*(t) + Y_{i+n}^*(t)$, it follows that

$$D = \int_0^\infty \sum_{i=1}^{2n} \{Z_i^* - \bar{Z}^*(t)\}\{Z_i^* - \bar{Z}^*(t)\}^\top Y_i^*(t)\mathrm{d}t, \qquad \text{and } d = \int_0^\infty \sum_{i=1}^{2n} \{Z_i^* - \bar{Z}^*(t)\}\mathrm{d}N_i^*(t).$$

Hence we can deal with delayed entry by replacing the original n observations with $2n$ pseudo observations and using the algorithms developed for the case where $S_1 = S_2 = \cdots = S_n = 0$.

The support for delayed entry is not only useful for implementing nonstandard time axes. It can also be used to implement (piecewise constant) time-varying explanatory variables, as well as to implement observations from more general counting processes.

4.2. Tuning parameter selection

A complete lasso or elastic net regularization path is useful mainly for judging relative importance of variables. In practice, the experimenter typically seeks the solution for a single value of the regularization parameter λ which can then be used similarly to how a set of unpenalized regression coefficients would be used. To select a 'representative' value of λ, cross-validation is commonly employed. As argued in Martinussen and Scheike (2009), the loss function (6) for the additive hazards model can be interpreted as a 'prediction error' within a quite general setting. It follows that if $F_1, \ldots, F_K$ is a partition of $\{1, \ldots, n\}$, each F_i being roughly the same size, we may define a cross-validation score

$$\mathrm{CV}(\lambda_l) := \sum_{i=1}^{K} L^{(F_i)}\{\hat{\beta}^{(-F_i)}(\lambda_l)\}, \quad l = 0, 1, \ldots, m; \tag{17}$$

with $L^{(F_i)}$ the loss calculated using observations from F_i only, and $\hat{\beta}^{(-F_i)}(\lambda_l)$ the penalized regression coefficients based on observations in $\{1, \ldots, n\} \backslash F_i$ only. We then select $\hat{\lambda} := \mathrm{argmin}_l \mathrm{CV}(\lambda_l)$ as the optimal λ-value.

In **ahaz**, 5-fold cross-validation ($K = 5$) is the default and offers an acceptable compromise between accuracy and stability in moderately sized data sets. In small data sets cross-validation can be somewhat unstable. For this reason, **ahaz** also supports repeated cross-validation where $\mathrm{CV}(\lambda)$ is averaged over several independent splits of $\{1, \ldots, n\}$ into folds. Observe that the cross-validation score in (17) does not support leave-one-out cross-validation (since $L^{(F_i)} \equiv 0$ when F_i is a singleton set).

It is also possible to select λ via criteria similar to BIC (or AIC). Although the loss function (6) is not based on a likelihood, we may still define the following analogue to BIC

$$\mathrm{PBIC}(\lambda) := \kappa L(\beta) + \mathrm{df}\{\hat{\beta}(\lambda)\} f(n); \tag{18}$$

where κ is some scaling constant. A convenient estimate of $\mathrm{df}\{\hat{\beta}(\lambda)\}$ is $\|\hat{\beta}(\lambda)\|_0$, the number of nonzero variables in $\hat{\beta}(\lambda)$ (Zou, Hastie, and Tibshirani 2007). Because the loss function

L is of the least-squares type, the arguments of Wang and Leng (2007) can be used to show that for p fixed, if $n^{-1}f(n) \to 0$ and $f(n) \to \infty$, the choice $\hat{\lambda} := \text{argmin}_\lambda \text{PBIC}(\lambda)$ entails certain selection consistency properties, depending on the underlying penalization method. For example, we can take $f(n) := \log n$ (Gorst-Rasmussen and Scheike 2011). In **ahaz**, we use

$$\kappa := \frac{d_{\mathcal{A}}^\top B_{\mathcal{A}}^- d_{\mathcal{A}}}{d_{\mathcal{A}}^\top D_{\mathcal{A}}^- d_{\mathcal{A}}};$$

where all quantities are calculated within the set $\mathcal{A}$ of nonzero variables at the smallest value of λ used, B is an estimator of the asymptotic covariance matrix of d, and X^- denotes the Moore-Penrose inverse. This choice of κ ensures that PBIC scales like a true BIC. Observe that (18), since it depends on the end point of the regularization path (through κ), is a sensible selection criterion primarily when $p < n$.

4.3. Survival predictions

The baseline cumulative hazard can be estimated by substituting a (penalized) estimator $\hat{\beta}$ in the definition of $\hat{\Lambda}_0$ in (2). This estimator is implemented in the `predict` routine in the **ahaz** package. Observe that $\hat{\Lambda}_0(t; \hat{\beta})$ is not guaranteed to be nondecreasing in t.

Survival predictions can be calculated as $\hat{S}(t|Z_i) = \exp(-\hat{\Lambda}_0(t; \hat{\beta}) - tZ_i^\top \hat{\beta})$. Considerable caution is necessary when working with such predictions since $\hat{S}(t|Z_i)$ need not be nonincreasing as a function of t and may even exceed 1. This issue of potentially 'non-physical' predictions is a known limitation of the additive hazards model and also of the more general nonparametric Aalen model (Aalen 1989). Lin and Ying (1994) suggest to 'monotonize' survival predictions by replacing $\hat{S}(t|Z_i)$ with $\min_{s \le t} \hat{S}(s|Z_i)$. Other monotonic regression methods such as the pooled-adjacent-violators algorithm could be applied instead. In practice, however, such post hoc correction is rarely ideal. Affecting primarily subjects with extreme risk scores $Z_i^\top \hat{\beta}$, it oftens leads to poor risk separation on the survival probability scale.

5. Timings and a data example

This section presents timing results for `ahazpen`, alongside an example of its usage on a real data set. We keep our timing study brief since previous work for other statistical models present a strong case that well-designed coordinate descent algorithms are universally faster than competing lasso fitting methods (Friedman *et al.* 2010; Simon *et al.* 2011).

5.1. Timings

Simon *et al.* (2011) used simulated data from a basic accelerated failure time model to assess runtimes of coordinate descent methods for the penalized Cox model. We adopt their simulation model for our runtime assessments and consider explanatory variables Z_i which are independent and identically distributed marginally standard Gaussian p-vectors satisfying $\text{Cor}(Z_{1j}, Z_{1k}) = \rho$ for $j \neq k$. True survival times are generated conditionally on the Z_is as

$$\tilde{T}_i := \exp\Big(\sum_{j=1}^{p} Z_{ij}\beta_j + W_i\Big), \quad i = 1, 2, \ldots, n$$

where $\beta_j := (-1)^j \exp(-2(j-1)/20)$ and W_i is a mean zero Gaussian random variable with variance such that the signal-to-noise ratio is 3.0. Censoring times are generated as $C_i := \exp(W_i)$ and the observed survival times as $T_i := \min(C_i, \tilde{T}_i)$.

We compare the runtime of `ahazpen` with that of `surv.lars` from the R package **timereg** (Scheike and Zhang 2011) which is currently the only publicly available software for fitting the lasso penalized additive hazards model. The `surv.lars` function is a modified version of the `lars` function from the package **lars** and requires pre-calculation of the quantities D and d. We use a highly efficient C-routine for calculating D based on formulas as in Section 3.1 (function `ahaz` in the package **ahaz**). To make `ahazpen` and `surv.lars` reasonably comparable, we stop `surv.lars` after 100 steps, and use the corresponding smallest λ-value $\lambda_{\min}$ to construct an exponentially decreasing λ-sequence for `ahazpen` of length 100 as in (11).

Experiments were run on an Intel Core I7 2.93 GHz, 8 GB RAM system with standard **BLAS**.

Runtimes for different values of n, p, and ρ are shown in Table 1 (averaged over 3 repetitions). Table 2 shows the corresponding runtimes of the pre-calculation part of lasso-LARS (averaged over the three repetitions and ρ as well). Coordinate descent is universally faster than lasso-LARS, especially for large values of p. The bottleneck of lasso-LARS is obviously the pre-calculation of D which has complexity of order $O(np^2)$. Neither algorithm is much affected by large correlations. Table 3 shows runtimes of `ahazpen` for very large values of n or p (averaged over 3 repetitions), based on a path of 100 λ-values with $\lambda_{\min}$ chosen such that the maximal number of variables in the path is roughly 100. It is seen that the algorithm is fully capable of dealing with large amounts of data. It runs more slowly for very large n than for very large p since the calculations in the strong/active sets become costly as well for large n. In our detailed assessments (not shown), the runtime scaled approximately linearly in n for fixed p and vice versa. Similar behavior was reported by Simon *et al.* (2011) for their coordinate descent algorithm for the penalized Cox model. Finally, a negative effect of large correlations on runtimes starts to become apparent for these large problems.

It is tempting to compare the raw computational performance of `ahazpen` with that of the **glmnet** `coxnet` function for fitting lasso penalized Cox models (Simon *et al.* 2011; Friedman *et al.* 2010). Such a comparison can only be qualitative and superficial since the algorithms solve different problems, use different convergence criteria, and rely on completely independent implementations. Intuitively, one might expect `ahazpen`, which is based on a linear model, to be substantially faster than `coxnet`. This is not the case. In fact, our limited experiments with **glmnet** (version 1.7) suggest that the two methods often have surprisingly similar runtimes, both being roughly as fast as **glmnet** coordinate descent for the simple linear regression model for an equally sized problem. A plausible explanation is that comparatively little time is spent on nonlinear optimizations because of the efficient use of active-set calculations. On the other hand, for very large n, `ahazpen` can be more efficient than `coxnet` since the coordinate descent part of `ahazpen` is essentially 'kernelized' (via the use of D, d); whereas `coxnet` does coordinate descent via inner products between n-vectors. Also, cross-validation tends to be somewhat faster for `ahazpen` than for `coxnet` since it is based on the simple quadratic loss (6).

An appreciable and implementation-independent advantage of the linearity of the additive hazards model is the guaranteed convergence `ahazpen`. It is also our experience that the runtime of `ahazpen` is more predictable than that of `coxnet` where convergence of the nonlinear optimization part can be sensitive to the nature of the data considered.

		$p = 100$		$p = 500$		$p = 5{,}000$		$p = 10{,}000$	
n	ρ	CCD	LAR	CCD	LAR	CCD	LAR	CCD	LAR
200	0	0.02	0.06	0.02	0.14	0.13	4.81	0.28	16.28
	0.25	0.02	0.07	0.02	0.15	0.12	4.62	0.28	16.13
	0.5	0.03	0.07	0.03	0.17	0.12	4.37	0.28	15.93
	0.9	0.06	0.07	0.04	0.14	0.13	4.29	0.28	15.48
	0.95	0.05	0.06	0.04	0.14	0.13	4.14	0.30	15.41
500	0	0.02	0.07	0.03	0.18	0.27	8.40	0.55	30.08
	0.25	0.03	0.07	0.04	0.18	0.29	8.15	0.56	29.79
	0.5	0.03	0.07	0.04	0.18	0.27	7.82	0.58	29.58
	0.9	0.04	0.07	0.04	0.18	0.28	7.86	0.55	29.65
	0.95	0.06	0.07	0.06	0.18	0.32	7.91	0.56	29.40
1,000	0	0.04	0.08	0.06	0.22	0.59	13.95	1.16	51.82
	0.25	0.04	0.07	0.06	0.22	0.55	13.80	1.06	51.52
	0.5	0.04	0.07	0.06	0.24	0.56	13.50	1.12	51.22
	0.9	0.05	0.07	0.06	0.22	0.61	13.50	1.14	51.07
	0.95	0.06	0.07	0.06	0.23	0.56	13.37	1.21	51.13

Table 1: Runtime (seconds) of `ahazpen` (CCD) and `surv.lars` (LAR) for the simulated data. Results are averaged over 3 repetitions.

	p			
n	100	500	5,000	10,000
200	0.00	0.03	3.32	13.47
500	0.00	0.06	6.80	27.05
1000	0.01	0.10	12.35	48.64

Table 2: Time (seconds) spent calculating D, d for the simulated data of Table 1 (averaged over 3 repetitions and ρ).

	ρ				
$(n; p)$	0	0.25	0.5	0.90	0.95
(200; 40,000)	1.22	1.19	1.19	1.18	1.30
(200; 100,000)	2.97	2.95	2.99	2.90	3.13
(200; 250,000)	6.84	6.84	6.79	6.91	7.13
(40,000; 200)	1.55	1.59	1.57	1.52	2.37
(100,000; 200)	4.03	3.99	3.96	3.89	6.02
(250,000; 200)	10.59	10.48	10.51	10.19	15.91

Table 3: Runtime (seconds) for `ahazpen` averaged over 3 repetitions.

5.2. An example using real data

To demonstrate the practical use of `ahazpen`, we consider the Sørlie data set (Sørlie *et al.* 2003) which consists of 549 gene expression measurements and survival times for 115 women diagnosed with breast cancer. This data set was also used by Martinussen and Scheike (2009) to demonstrate the lasso penalized additive hazards model.

We consider the challenging problem of performing lasso penalized survival regression for both main effects and pairwise (multiplicative) interactions of gene expressions. The design matrix has $p = 549 + 549 \cdot (549 - 1)/2 = 150,975$ columns. We apply the additive hazards lasso directly to this design matrix, ignoring here the discussion whether it is sensible to allow for inclusion of interactions without the corresponding main effects.

We load and format the data as follows (note that generating `X` may take several minutes):

```
R> data("sorlie")
R> set.seed(10101)
R> surv <- Surv(sorlie$time + runif(nrow(sorlie)) * 1e-2, sorlie$status)
R> Z <- sorlie[, 3:ncol(sorlie)]
R> p <- ncol(Z)
R> pw.comb <- combn(1:p, 2)
R> X <- cbind(Z, Z[, pw.comb[1, ]] * Z[, pw.comb[2, ]])
```

It is common practice to put variables on the same scale before applying the lasso. In `ahazpen`, data is scaled by default (estimates are returned on the original scale) so it is not necessary standardize data manually. We make the following call to `ahazpen` to fit the lasso; the choice of `penalty` corresponds to the default value and is included here for completeness:

```
R> fit.init <- ahazpen(surv, X, dfmax = 50,
+    penalty = lasso.control(alpha = 1))
R> fit.init

Call:
ahazpen(surv = surv, X = X, dfmax = 50)

* No. predictors:              150975
* No. observations:               115
* Max no. predictors in path:      53
* Penalty parameter lambda:
    -No. grid points:      32
    -Min value:        0.1057
    -Max value:        0.2700
```

To prevent `ahazpen` from calculating a complete regularization path, `dfmax` has been specified. This option is useful for reducing computation time since, in practice, the lasso often prefers rather sparse solutions. Only 32 λ-values are used even though `ahazpen` is set to use 100 λ-values as default. This is because `ahazpen` cannot anticipate the λ-value at which `dfmax` is reached and hence simply truncates the default λ-sequence (11). A grid of λ-values with the desired density is easily obtained by a second call to `ahazpen`:

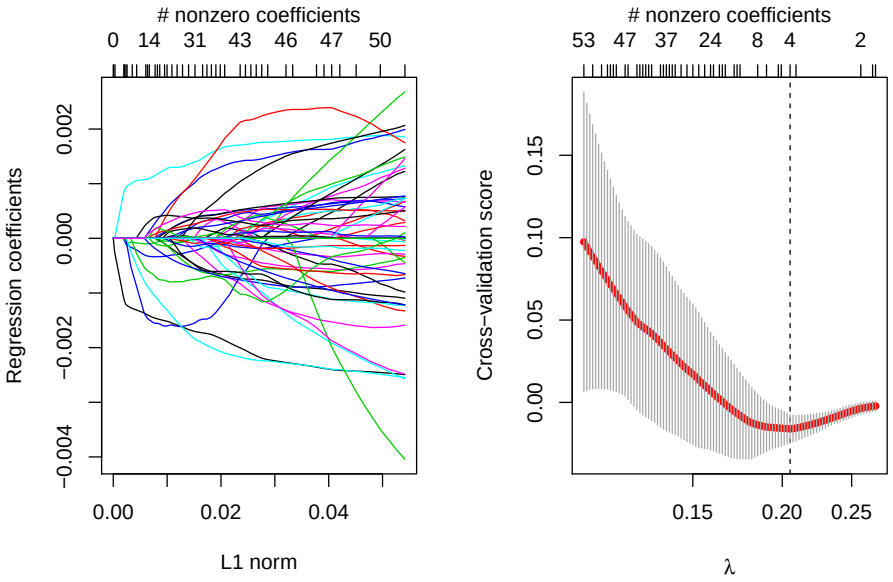

Figure 1: Plot of regularization path (left) and 5-fold cross-validation scores (right) for the additive hazards lasso applied to the Sørlie gene expression data with main effects and pairwise interactions.

```
R> l <- range(fit.init$lambda)
R> fit <- ahazpen(surv, X, lambda.minf = l[1]/l[2])
R> plot(fit)
```

Evaluating this `ahazpen` call took roughly 9 seconds. A plot of the regularization path is shown in Figure 1 (left).

To determine an optimal value of λ, we use 5-fold cross-validation as follows:

```
R> set.seed(10101)
R> fit.tune <- tune.ahazpen(surv, X, lambda.minf = l[1]/l[2], tune = "cv")
R> fit.tune

Call:
tune.ahazpen(surv = surv, X = X, tune = "cv", lambda.minf = l[1]/l[2])

Cross-validation:  5  folds

Length of lambda sequence :  100
Optimal lambda            :  0.2051
d.f. at optimal lambda    :  4

R> plot(fit.tune)
```

Typically, K-fold cross-validation takes about as long as running `ahazpen` $K + 1$ times. Figure 1 (right) shows the curve of cross-validation scores. Indices of the final nonzero regression coefficients are then obtained as follows

```
R> beta <- coef(fit.tune)
R> which(as.numeric(beta) != 0)

[1]  21 269 346 401
```

Apparently, the lasso prefers a model containing main effects only.

6. Discussion

Cyclic coordinate descent is a simple numerical optimization method which works exceptionally well for penalized regression problems with variable selection. We have developed a coordinate descent algorithm for the elastic net penalized additive hazards model and provided an implementation via the `ahazpen` function in the R package **ahaz**. This function can handle very large amounts of data efficiently and is an important and flexible alternative to the more commonly used elastic net penalized Cox model. In terms of computational properties, the additive hazards model is intrinsically linear which implies theoretically guaranteed convergence of `ahazpen` and highly predictable runtimes in practice. Our specific implementation provides support for survival data with delayed entry which in turn enables the use of more complex data types such as nonstandard time axes, time-varying covariates, and general counting process data.

Acknowledgments

Part of Anders Gorst-Rasmussen's research was carried out while employed at Center for Cardiovascular Research, Aalborg Hospital, Aarhus University Hospital, Denmark.

References

Aalen OO (1989). "A Linear Regression Model for the Analysis of Life Times." *Statistics in Medicine*, **8**, 907–925.

Efron B, Hastie T, Johnstone I, Tibshirani R (2004). "Least Angle Regression." *The Annals of Statistics*, **32**, 407–499.

Fan J, Li R (2001). "Variable Selection via Nonconcave Penalized Likelihood and Its Oracle Properties." *Journal of the American Statistical Association*, **96**, 1348–1360.

Friedman J, Hastie T, Höfling H, Tibshirani R (2007). "Pathwise Coordinate Optimization." *The Annals of Applied Statistics*, **1**, 302–332.

Friedman J, Hastie T, Tibshirani R (2010). "Regularization Paths for Generalized Linear Models via Coordinate Descent." *Journal of Statistical Software*, **33**(1), 1–22. URL http://www.jstatsoft.org/v33/i01/.

Goeman JJ (2010). "L1 Penalized Estimation in the Cox Proportional Hazards Model." *Biometrical Journal*, **52**, 70–84.

Gorst-Rasmussen A, Scheike T (2011). "Independent Screening for Single-index Hazard Rate Models with Ultra-high Dimensional Features." *Technical Report R-2011-06*, Department of Mathematical Sciences, Aalborg University. URL http://arxiv.org/abs/1105.3361.

Gui J, Li H (2005). "Penalized Cox Regression Analysis in the High-Dimensional and Low-Sample Size Settings, with Applications to Microarray Gene Expression Data." *Bioinformatics*, **21**, 3001–3008.

Leng C, Ma S (2007). "Path Consistent Model Selection in Additive Risk Model via Lasso." *Statistics in Medicine*, **26**, 3753–3770.

Li H (2008). "Censored Data Regression in High-Dimensional and Low-Sample-Size Settings for Genomic Applications." In A Biswas, S Datta, J Fine, M Segal (eds.), *Statistical Advances in Biomedical Sciences: State of the Art and Future Directions*. John Wiley & Sons.

Lin DY, Ying Z (1994). "Semiparametric Analysis of the Additive Risk Model." *Biometrika*, **81**, 61–71.

Ma S, Huang J, Shi M, Li Y, Shia B (2010). "Semiparametric Prognosis Models in Genomic Studies." *Briefings in Bioinformatics*, **11**, 385–393.

Ma S, Kosorok M, Fine JP (2006). "Additive Risk Models for Survival Data with High-Dimensional Covariates." *Biometrika*, **62**, 202–210.

Martinussen T, Scheike TH (2009). "Covariate Selection for the Semiparametric Additive Risk Model." *Scandinavian Journal of Statistics*, **36**, 602–619.

Martinussen T, Scheike TH (2010). "The Additive Hazards Model with High-Dimensional Regressors." *Lifetime Data Analysis*, **15**, 330–342.

McKeague IW, Sasieni PD (1994). "A Partly Parametric Additive Risk Model." *Biometrika*, **81**, 501–514.

Park MY, Hastie T (2007). "L1 Regularization Path Algorithm for Generalized Linear Models." *Journal of the Royal Statistical Society B*, **69**, 659–677.

R Development Core Team (2012). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org/.

Scheike TH, Zhang MJ (2011). "Analyzing Competing Risk Data Using the R **timereg** Package." *Journal of Statistical Software*, **38**, 1–15. URL http://www.jstatsoft.org/v38/i02/.

Simon N, Friedman J, Hastie T, Tibshirani R (2011). "Regularization Paths for Cox's Proportional Hazards Model via Coordinate Descent." *Journal of Statistical Software*, **39**(5), 1–13. URL http://www.jstatsoft.org/v39/i05/.

Sohn I, Kim J, Jung S, Park C (2009). "Gradient Lasso for Cox Proportional Hazards Model." *Bioinformatics*, **25**, 1775–1781.

Sørlie T, Partker R, Hatie T, Marron J, Nobel A, Deng S, Johnsen H, Pesich R, Geisler S, Demeter J, Peour C, Lønning P, Brown P, Børresen-Dale A, Botstein D (2003). "Repeated Observation of Breast Tumor Subtypes in Independent Gene Expression Data Sets." *Proceedings of the National Academy of Sciences*, **100**, 8418–8423.

Tibshirani R (1997). "The Lasso Method for Variable Selection in the Cox Model." *Statistics in Medicine*, **16**, 385–395.

Tibshirani R, Bien J, Friedman J, Hastie T, Simon N, Taylor J, Tibshirani RJ (2012). "Strong Rules for Discarding Predictors in Lasso-Type Problems." *Journal of the Royal Statistical Society B*, **74**(2), 245–266.

Tseng P (1988). "Coordinate Ascent for Maximizing Nondifferentiable Concave Functions." *Technical Report LIDS-P, 1840*, Massachusetts Institute of Technology, Laboratory for Information and Decision Systems.

Wang H, Leng C (2007). "Unified LASSO Estimation by Least Squares Approximation." *Journal of the American Statistical Association*, **102**, 1039–1048.

Zou H (2006). "The Adaptive Lasso and Its Oracle Properties." *Journal of the American Statistical Association*, **101**, 1418–1429.

Zou H, Hastie T (2005). "Regularization and Variable Selection via the Elastic Net." *Journal of the Royal Statistical Society B*, **67**, 301–320.

Zou H, Hastie T, Tibshirani R (2007). "On the "Degrees of Freedom" of the Lasso." *The Annals of Statistics*, **35**, 2173–2192.

Affiliation:

Anders Gorst-Rasmussen
Department of Mathematical Sciences
Frederik Bajers Vej 7G
9220 Aalborg East, Denmark
E-mail: anders@gorst.dk
URL: http://www.gorst.dk/

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume 47, Issue 9 *Submitted:* 2011-07-05
April 2012 *Accepted:* 2012-03-06