pcox Package

Jonathan Gellar

1 Introduction

pcox is an R program to fit penalized Cox models with smooth effects of covariates, using a penalized spline basis. These effects can either be smooth over the domain of the covariate, smooth over the time domain, or both. Both scalar and functional predictors are supported. Functional predictors can either be measured along their own domain, or along the same time domain as the hazard function. We refer to the former type of predictor as a static functional covariate, and the latter as a time-varying covariate.

These definitions result in the following general model:

$$\log \lambda_i(t) = \log \lambda_0(t) + f(x_i, t) + g(X_i, t) + h(Z_i, t) \tag{1}$$

In this notation $\lambda_i(t)$ is the hazard function for subject i and $\lambda_0(t)$ is the baseline hazard function; x_i, X_i, and Z_i are scalar, static functional, and time-varying functional covariates, respectively; and f, g, and h are unknown functions we want to estimate. As proposed above this model is very general, but in practice we will likely want to parameterize the way in which each covariate affects the hazard function. Table 1 summarizes the different parameterizations for f, g, and h, and the corresponding syntax for the pcox formula. These terms will be discussed further in Section 3.

Table 1: Summary of the types of terms allowed in an `pcox` formula.

Time Effect	Covariate Effect	Term Formula	Implementation in `pcox`
Scalar Predictors:			
Static	Linear	βx	`x`
Static	Nonlinear	$\beta(x)$	`s(x)`
Varying	Linear	$\beta(t)x$	`tv(x)`
Varying	Nonlinear	$\beta(x,t)$	`tv(s(x))`
Baseline Functional Predictors:			
Static	Linear	$\int_{\mathcal{S}} \beta(s)X_i(s)\,ds$	`lf(X)`, `lf.vd(X)`
Static	Nonlinear	$\int_{\mathcal{S}} \beta[s, X_i(s)]\,ds$	`af(X)`
Varying	Linear	$\int_{\mathcal{S}} \beta(s,t)X_i(s)\,ds$	`tv(lf(X))`, `tv(lf.vd(X))`
Varying	Nonlinear	$\int_{\mathcal{S}} \beta[s,t,X_i(s)]\,ds$	`tv(af(X))`
Concurrent Functional Predictors:			
Static	Linear	$\beta Z_i(t)$	`Z` or `clf(Z)`
Static	Nonlinear	$\beta[Z_i(t)]$	`s(Z)` or `caf(Z)`
Varying	Linear	$\beta(t)Z_i(t)$	`tv(Z)` or `tv(clf(Z))` or `clf(Z, tv=TRUE)`
Varying	Nonlinear	$\beta[Z_i(t),t]$	`tv(caf(Z))` or `caf(Z, tv=TRUE)`
Historical Functional Predictors:			
Static	Linear	$\int_{\delta(t)}^{t} \beta(s)Z_i(s)\,ds$	`hlf(Z)`
Static	Nonlinear	$\int_{\delta(t)}^{t} \beta[s, Z_i(s)]\,ds$	`haf(Z)`
Varying	Linear	$\int_{\delta(t)}^{t} \beta(s,t)Z_i(s)\,ds$	`tv(hlf(Z))` or `hlf(Z, tv=TRUE)`
Varying	Nonlinear	$\int_{\delta(t)}^{t} \beta[s,t,Z_i(s)]\,ds$	`tv(haf(Z))` or `haf(Z, tv=TRUE)`
Random Effects:			

The basic mechanism for modeling the nonparametric effects is to apply a penalized spline basis to the functions f, g, and h in (1). The model is then estimated by maximizing the penalized partial likelihood (PPL),

$$\ell_\rho^{(p)}(\boldsymbol{\theta}) = \sum_{i:\delta_i=1} \left\{ \eta_i(\boldsymbol{\theta}, t) - \log \left(\sum_{j:Y_j \geq Y_i} e^{\eta_j(\boldsymbol{\theta},t)} \right) \right\} - \rho P(\boldsymbol{\theta}) \tag{2}$$

where $\boldsymbol{\theta}$ are the model parameters (e.g., spline coefficients), $\eta_i(\boldsymbol{\theta}, t) = f(x_i, t) + g(X_i, t) + h(Z_i, t)$ is the linear predictor for subject i at time t, ρ is a smoothing parameter (possibly a vector), and $P(\boldsymbol{\theta})$ is an appropriate penalty on the parameters. For a given ρ, the PPL is maximized

by Newton-Raphson. The smoothing parameter(s) ρ may be optimized by a number of different criteria: the cross-validated likelihood (CVL), a likelihood-based information criterion (AIC, AIC$_c$, or EPIC), maximum likelihood (ML), or restricted maximum likelihood (REML). We intend to make each of these optimization criteria available in `pcox`, except for the CVL, which is quite computationally intensive.

`pcox` is essentially a wrapper for four other `R` packages: `survival` and `coxme` for fitting penalized Cox models, and `mgcv` and `refund` for processing the covariates. Section 2 describes how the models are fit, and Section 3 describes how the covariates are processed.

2 Model Fitting

2.1 Basic structure

We take advantage of existing implementations of penalized Cox models in `R`. Two functions that fit these models are the `coxph()` function from the `survival` package, and the `coxme()` function from the package of the same name. The former optimizes the smoothing parameter using a user-defined control function, whereas the latter is optimized specifically through ML or REML. `pcox` is essentially a wrapper for these two functions. If the user chooses to optimize the model via ML or REML, `coxme()` is called, otherwise `coxph()` is called.

2.2 Specifying penalized terms in a `coxph` formula

Instructions for defining penalized terms in a `coxph` formula are found in **?**. The idea is to create a `coxph.penalty` object, which is the model matrix corresponding to the term, with a number of attributes defining the penalization. These attributes include a function to compute the first and second derivatives of the penalty, the control function for the "outer" loop to optimize the smoothing parameter, and parameters for these functions. I have created the function `acTerm()` to convert an `mgcv` smooth term (created by calling `smoothCon()` into a `coxph.penalty` object.

2.3 Specifying penalized terms in a `coxme` formula

`coxme` was designed to fit Gaussian frailty models, where the user enters the distribution of the random effects by supplying a variance matrix. This variance corresponds to the inverse of the penalty matrix from a penalized Cox model. Unfortunately, many of the common penalty matrices that we use are non-invertible. I am currently working with the developer of `coxme`, Terry Therneau, to allow `coxme()` to accept the penalty matrix instead of the variance matrix.

3 Covariate Processing

Penalized spline bases are implemented for generalized additive models very flexibly by the `mgcv` package; we take advantage of this flexibility by allowing for `mgcv`-style model terms. This allows the choice of basis, number of knots, and form of the penalty to be selected by the user.

Terms involving functional predictors can be specified using certain terms from the `refund` package, which creates the appropriate `mgcv`-style regression term.

Time varying covariates can either be included as concurrent terms or historical terms. Concurrent terms allow only the value of the covariate at time t to impact the hazard at time t; this is the traditional way of handling time-varying covariates in a Cox model. Historical terms allow the hazard at time t to depend on the entire history of the covariate up to and including time t.

3.1 Time-varying effects

Time varying effects can be specified as a `tv()` term.

4 Examples

I was just getting down to implementing the code for concurrent functional predictors. Recall that my plan was to have these terms indicated just as a matrix in the model formula, i.e.

‘Surv(time,event) Z‘ for a matrix ‘Z‘. However, this doesn’t let us specify the time points that correspond to the columns of ‘Z‘ - it requires making an assumption on how these columns relate to the ‘time‘ variable (i.e., there needs to be one column for each time point). It also doesn’t allow for any other options regarding how ‘Z‘ is processed - I can’t think of any of these right now that we might want, but it may come up.

A much more flexible way of specifying these terms would be as a function, e.g. ‘clf(Z)‘ for a "concurrent linear function" and ‘caf(Z)‘ for a "concurrent additive function". This would correspond to how we allow for historical terms with ‘hlf()‘ and ‘haf()‘.

Another option is to drop the "l" and "a" for all of these functions. So we would just have ‘hf()‘ and ‘cf()‘, both of which would have an argument ‘additive‘ which defaults to ‘FALSE‘.