Supplemental Material For:

Cox Regression Models with Functional Covariates for Survival Data

Contents

1 Software Implementation

Our methods have been implemented as part of the `pcox` software package in the `R` computing environment. This code is a wrapper for three very popular and well-tested software packages: `refund` (Crainiceanu et al., 2012) for processing functional predictors, `mgcv` (Wood, 2006) for setting up the basis and penalization, and `survival` (Therneau and Grambsch, 2000; Therneau, 2014) for estimation via the penalized partial likelihood. We expect to release a publicly available version of the software on CRAN in late 2014; email the corresponding author if you are interested in the beta version before this release.

For details on the inner mechanics of setting up a user-defined penalized term using `coxph()`, see Therneau and Grambsch (1998). Using this structure, we have built the `pcox()` function, which transparently fits penalized Cox models by maximizing the penalized partial likelihood. By using `coxph()` to perform the model estimation and returning a `coxph` object, we enable the user to take full advantage of the existing features of this highly efficient and well-tested function. It also allows the code to be of a familiar format, lessening the learning curve for the end user. Models may be fit using just a single line of code.

`pcox` is able to process both functional and scalar terms. Functional terms are incorporated using `refund::lf()` for linear functional terms, `refund::af()` for additive functional terms (McLean et al., 2012), or `refund::lf.vd()` for variable-domain linear functional terms (Gellar et al., 2014). Smooth effects of scalar covariates may be incorporated using `mgcv::s()` or `mgcv::te()`. All of these functions are flexible in that they allow for different basis choices, knots, and penalization schemes, among other options.

The focus of this paper is on Cox regression with penalized linear functional terms, which is accomplished by incorporating a `lf()` term. The only required argument of the function is `X`, an $N \times J$ matrix of functional predictors, where N is the number of subjects and J is the number of observation points per curve. The following code is all that is required to fit a Cox regression with functional predictor `X`:

```
fit <- pcox(Surv(Y,delta) ~ lf(X), data=mydata)
```

The left hand side of the formula uses syntax from the `survival` package to create a survival object. The above line of code, for $N = 500$ subjects and $J = 101$ observations per curve, fits in less than 0.5 seconds on a common laptop (MacBook Pro, 2GHz processor, 16 GB RAM). The observation points $\{s_j\}$ may also be entered, though this argument defaults to equally spaced observations between 0 and 1. The functional predictor may be modeled using any basis supported by `mgcv`, with all the associated options for the form of the basis and penalty.

Another advantage of using `coxph()` to maximize the penalized partial likelihood is that the authors already have designed a very efficient algorithm to optimize the smoothing parameter based on either the AIC or AIC_c, using Brent's method (Brent, 1973). We modified this code slightly to also allow the user to specify EPIC-based optimization. Alternatively, either the value of the smoothing parameter or the desired degrees of freedom for the term may be entered to fix λ at a particular value. Additional options include the ability to specify the convergence level for the smoothing parameter, the numerical integration method, the number of basis functions, and a flag for pre-smoothing the functions using a functional principal component decomposition, thereby allowing `X` to contain missing observations. A slightly more complex model may be fit as

```
fit <- pcox(Surv(Y, delta) ~ age + sex + lf(X, s=s.vec, presmooth=TRUE,
            integration="simpson") + s(log.los, bs="ps", k=15),
            method="epic", eps=0.00001), data=mydata)
```

Here `age` and `sex` are scalar covariates, and `log.los` is incorporated as a smooth p-spline term. A functional principal components decomposition is performed on the `X` matrix, numerical integration is done using Simpson's method, and λ is optimized by minimizing EPIC to the nearest 0.00001. The model fits in approximately 2.5 seconds, with the primary increase in computation time due to adding a principal components decomposition step to smooth the functional data.

2 Additional Simulation Results

2.1 Effect of pre-smoothing with FPCA

In the main text, we compared simulation results for the case when there is no missing data, to when there is missing data. The case with missing data requires the functions to be pre-smoothed using a functional principal components basis. In order to isolate the effect of the FPCA step, we compare the results with and without pre-smoothing, using the full data for both cases.

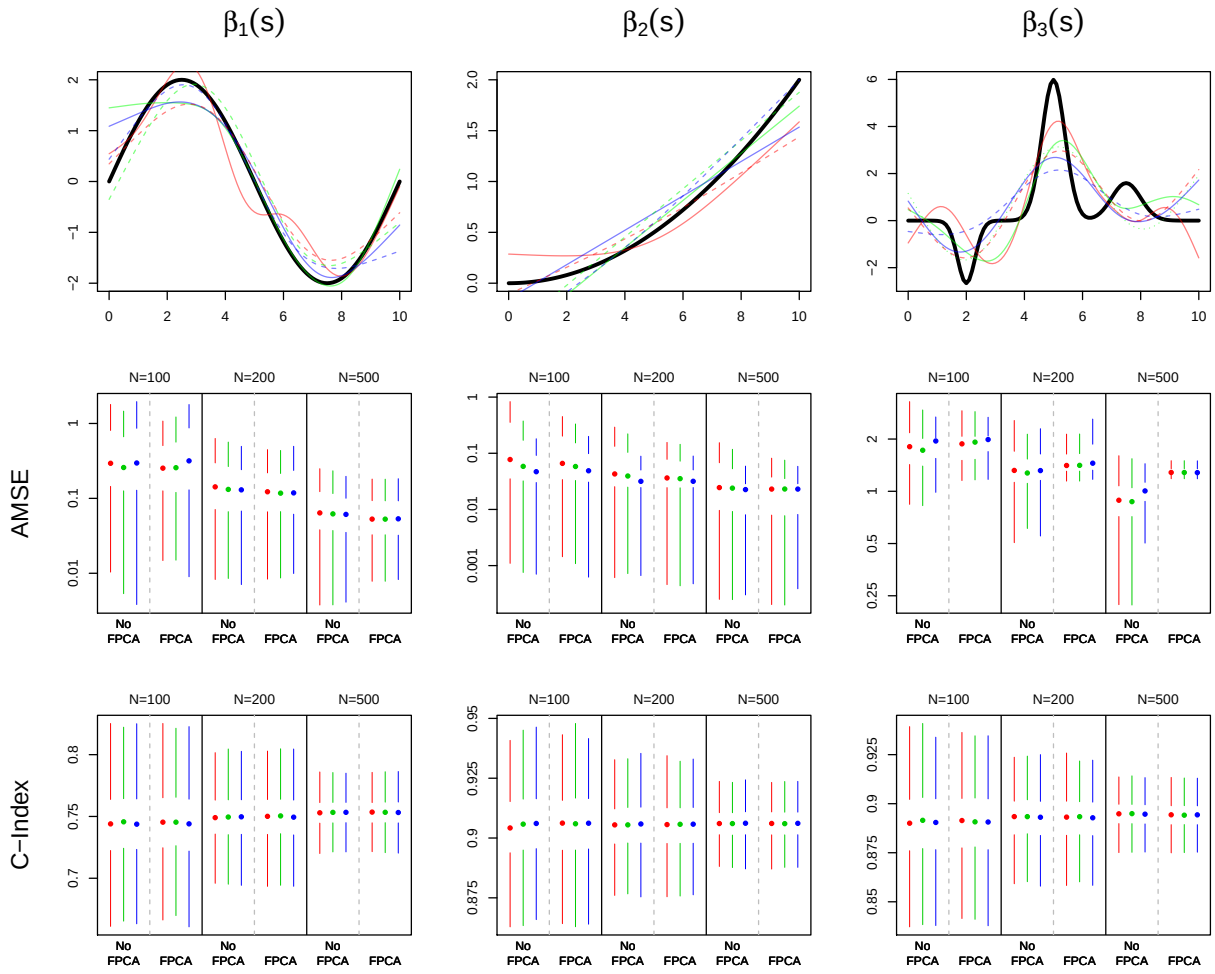

Figure 1: Simulation results for the full data, comparing with vs. without pre-smoothing using an FPCA basis. Plots are similar in style to Figure 1 of the main text.

2.2 Coverage probabilities vs. s

In this section, we examine how the coverage probabilities vary with the domain width s in our simulated data. We present only the results for moderate sample size ($N = 200$), for each of the three optimization criteria.

AIC criterion:

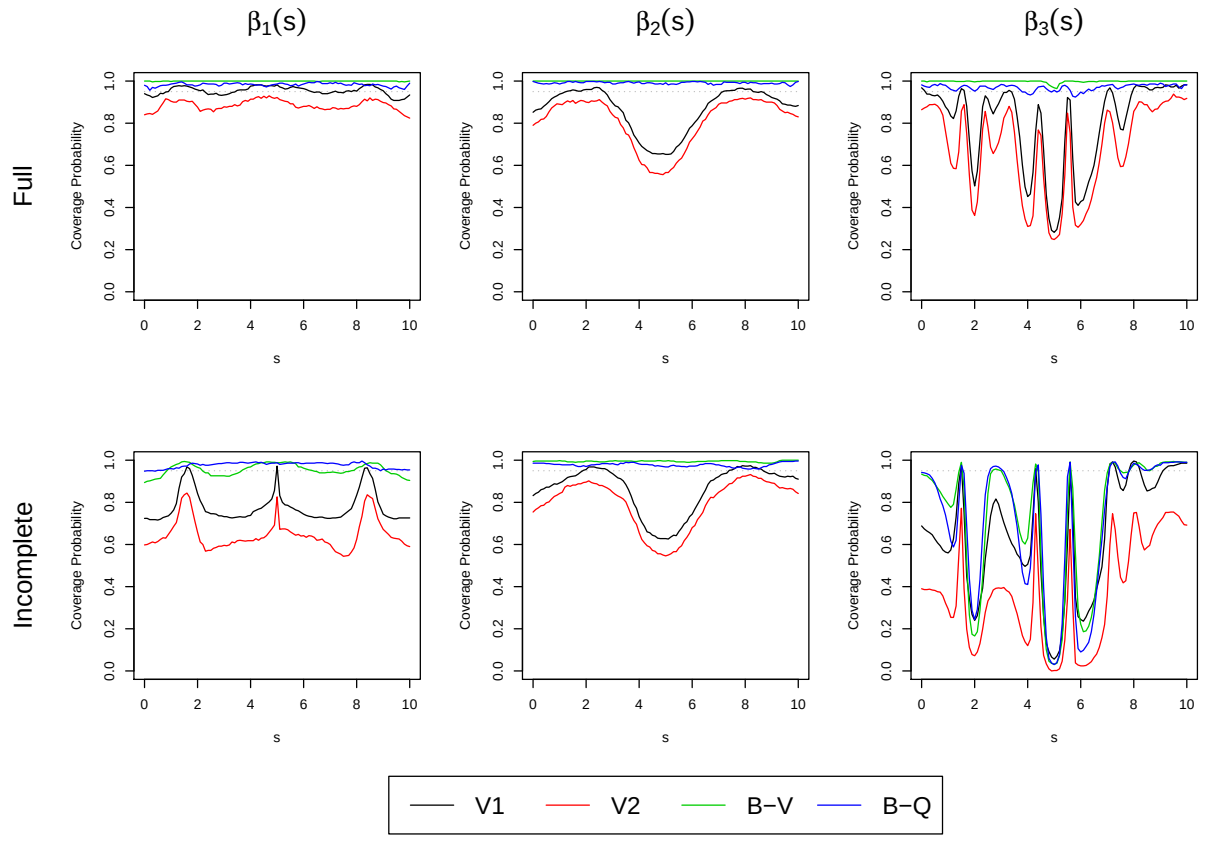

Figure 2: Coverage probabilities of each of the four point wise 95% confidence intervals vs. s, for $N = 200$ and the AIC criterion. V1 and V2 are Wald-type confidence intervals based on the model-based estimates of the variance V_1 and V_2, defined in Section 2.4 of the main text. B-V is a Wald-type confidence interval based on the variance of the bootstrap estimates, and B-Q is constructed from the 2.5% and 97.5% quantiles of the boostrap distribution.

$\mathbf{AIC}_c$ criterion:

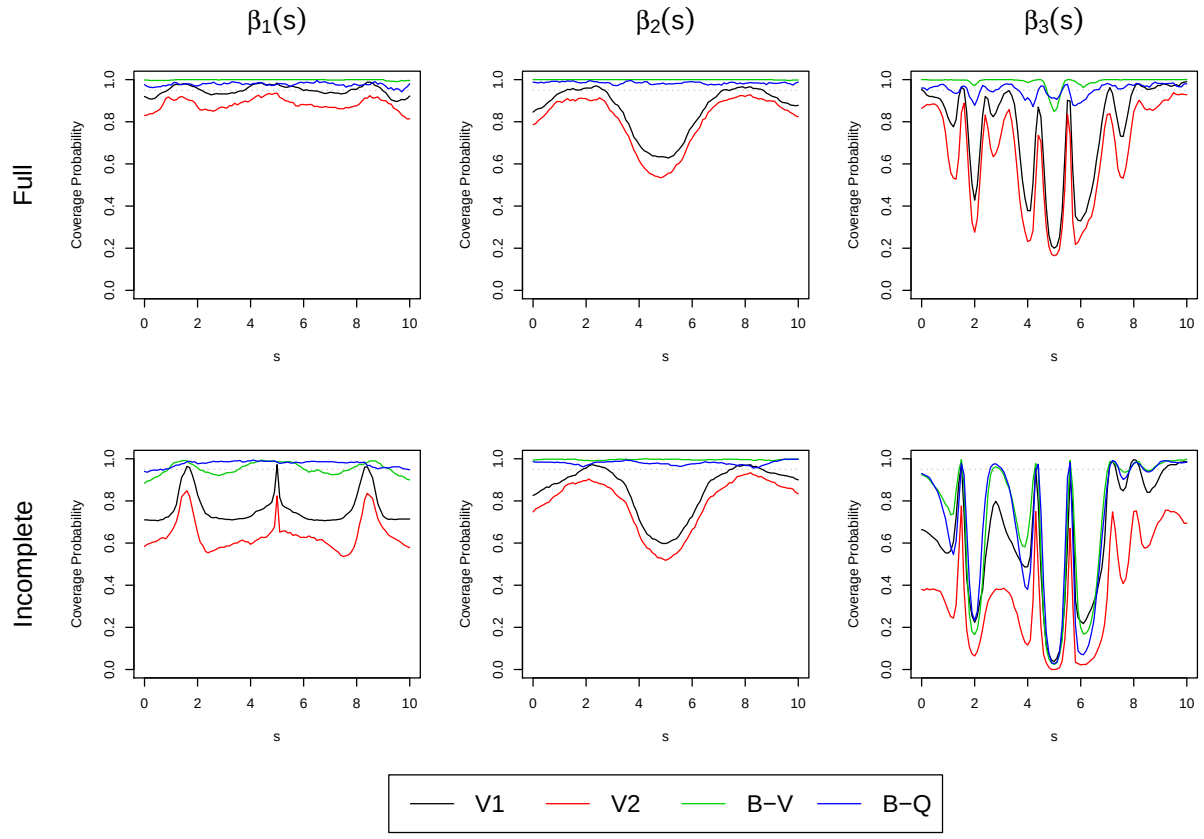

Figure 3: Coverage probabilities of each of the four point wise 95% confidence intervals vs. s, for $N = 200$ and the AIC_c criterion. V1 and V2 are Wald-type confidence intervals based on the model-based estimates of the variance V_1 and V_2, defined in Section 2.4 of the main text. B-V is a Wald-type confidence interval based on the variance of the bootstrap estimates, and B-Q is constructed from the 2.5% and 97.5% quantiles of the boostrap distribution.

EPIC criterion:

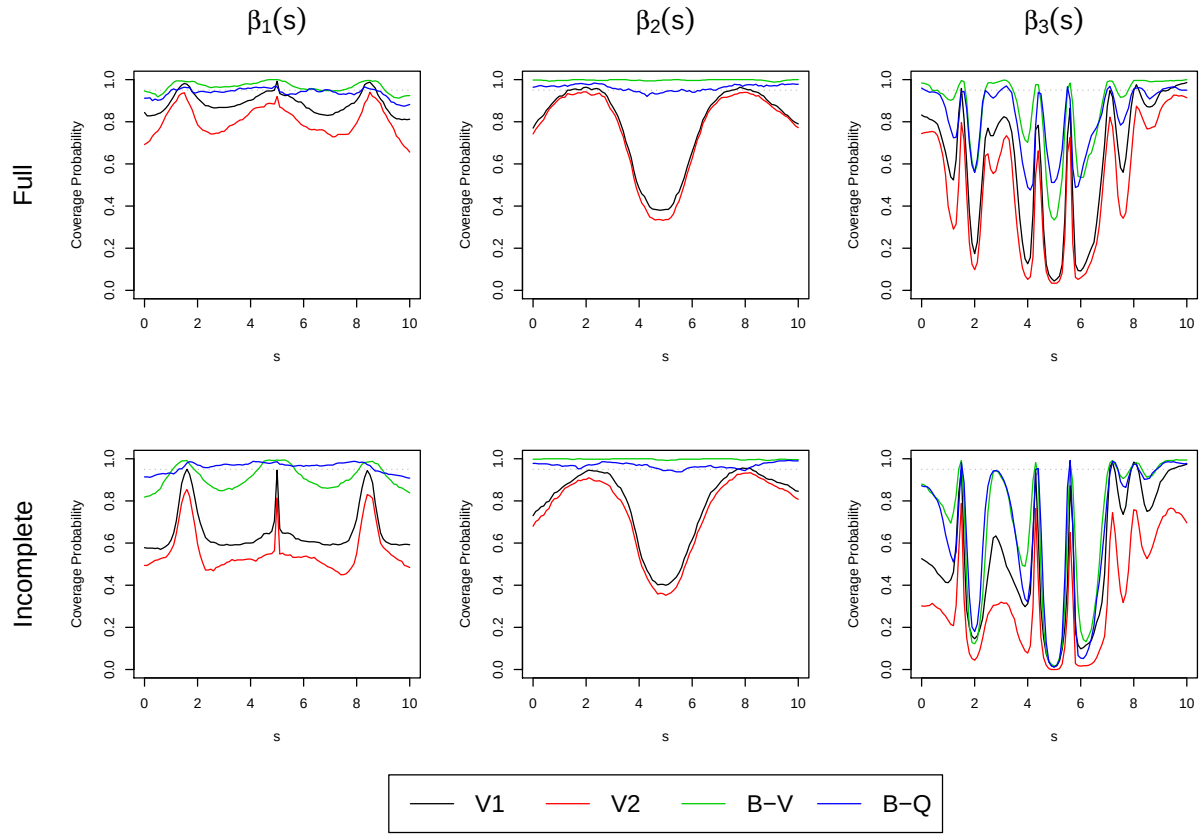

Figure 4: Coverage probabilities of each of the four point wise 95% confidence intervals vs. s, for $N = 200$ and the EPIC criterion. V1 and V2 are Wald-type confidence intervals based on the model-based estimates of the variance V_1 and V_2, defined in Section 2.4 of the main text. B-V is a Wald-type confidence interval based on the variance of the bootstrap estimates, and B-Q is constructed from the 2.5% and 97.5% quantiles of the boostrap distribution.

References

Brent, R. (1973). Chapter 4. In *Algorithms for minimization without derivatives*. Prentice-Hall, Englewood Cliffs, NJ.

Crainiceanu, C., Reiss, P., Goldsmith, J., Huang, L., Huo, L., Scheipl, F., Greven, S., Harezlak, J., Kundu, M., and Zhao, Y. (2012). refund: Regression with Functional Data, R package version 0.1-6.

Gellar, J. E., Colantuoni, E., Needham, D. M., and Crainiceanu, C. M. (2014). Variable-Domain Functional Regression for Modeling ICU Data (in-press). *Journal of the American Statistical Association*.

McLean, M. W., Hooker, G., Staicu, A.-M., Scheipl, F., and Ruppert, D. (2012). Functional Generalized Additive Models. *Journal of Computational and Graphical Statistics*, (May 2013):130322113058003.

Therneau, T. M. (2014). A Package for Survival Analysis in S.

Therneau, T. M. and Grambsch, P. M. (1998). Penalized Cox models and Frailty. *Technical report, Division of Biostatistics. Mayo Clinic; Rochester, MN*, pages 1–58.

Therneau, T. M. and Grambsch, P. M. (2000). *Modeling survival data: extending the Cox model*. Springer, New York.

Wood, S. (2006). *Generalized additive models: an introduction with R*, volume 66. Chapman & Hall/CRC.