How to use the `coxed` function

Jonathan Kropko and Jeffrey J. Harden

2020-08-01

1 Introduction

The Cox proportional hazards model (implemented in R as `coxph()` in the `survival` package or as `cph()` `rms` package) is one of the most frequently used estimators in duration (survival) analysis. Because it is estimated using only the observed durations' rank ordering, typical quantities of interest used to communicate results of the Cox model come from the hazard function (e.g., hazard ratios or percentage changes in the hazard rate). These quantities are substantively vague and difficult for many audiences of research to understand. The `coxed` package introduces a suite of methods to address these problems. The package allows researchers to calculate duration-based quantities from Cox model results, such as the expected duration (or survival time) given covariate values and marginal changes in duration for a specified change in a covariate. These duration-based quantities often match better with researchers' substantive interests and are easily understood by most readers. This document is a walkthrough of the examples included in the help documentation for the `coxed()` function.

Before we begin, we load the `coxed` package,

```
library(coxed)
```

and packages from the `tidyverse` for managing and plotting data as we go:

```
library(dplyr)
library(tidyr)
library(ggplot2)
```

2 Running a Cox proportional hazards model

The following quote from Kropko and Harden (2018) sets up our first working example:

> Martin and Vanberg (2003) examine the determinants of negotiation time among political parties forming a coalition government. . . . The dependent variable in Martin and Vanberg's analysis is the number of days between the beginning and end of the bargaining period. Martin and Vanberg model this variable as a function of the Range of government, which is a measure of the ideological distance between the extreme members of the coalition, the Number of government parties in the coalition, and several other variables. They interact Number of government parties with the natural log of time because that variable violates the proportional hazards assumption. Their hypotheses predict negative coefficients on the variables of interest, indicating that increases in the ideological distance between the parties and in the number of parties correspond with a decrease in the risk of government formation, or a longer negotiation time.

The authors demonstrate support for their hypotheses by computing changes in the hazard rate based on changes to these independent variables. However, to assess what the estimated effects of Range of government and Number of government parties mean in substantive terms, we use `coxed()` to predict how long is each case predicted to last. We will also find answers to the following questions about duration:

- How much longer will negotiations take for an ideologically polarized coaltion as compared to an ideologically homogeneous one?
- How much longer will negotiations take for a multiparty coalition government than for a single-party government?

First we replicate the Cox model from Martin and Vanberg (2003):

```
mv.surv <- Surv(martinvanberg$formdur, event = rep(1, nrow(martinvanberg)))
mv.cox <- coxph(mv.surv ~ postel + prevdef + cont + ident + rgovm + pgovno +
                     tpgovno + minority, method = "breslow", data = martinvanberg)
summary(mv.cox)
```

```
## Call:
## coxph(formula = mv.surv ~ postel + prevdef + cont + ident + rgovm +
##     pgovno + tpgovno + minority, data = martinvanberg, method = "breslow")
##
##   n= 203, number of events= 203
##
##              coef exp(coef) se(coef)       z Pr(>|z|)
## postel   -0.57665   0.56177  0.16862  -3.420 0.000627 ***
## prevdef  -0.10000   0.90484  0.22987  -0.435 0.663543
## cont      1.10047   3.00556  0.23970   4.591 4.41e-06 ***
## ident     0.14579   1.15695  0.11859   1.229 0.218938
## rgovm    -0.21312   0.80806  0.12036  -1.771 0.076625 .
## pgovno    1.19057   3.28894  0.12405   9.598  < 2e-16 ***
## tpgovno  -0.43189   0.64928  0.03476 -12.425  < 2e-16 ***
## minority -0.42759   0.65208  0.20793  -2.056 0.039740 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##          exp(coef) exp(-coef) lower .95 upper .95
## postel      0.5618     1.7801    0.4037    0.7818
## prevdef     0.9048     1.1052    0.5766    1.4198
## cont        3.0056     0.3327    1.8789    4.8079
## ident       1.1570     0.8643    0.9170    1.4597
## rgovm       0.8081     1.2375     0.6382     1.0231
## pgovno      3.2889     0.3040     2.5791     4.1942
## tpgovno     0.6493     1.5402     0.6065     0.6951
## minority    0.6521     1.5336     0.4338     0.9801
##
## Concordance= 0.903  (se = 0.007 )
## Likelihood ratio test= 277.2  on 8 df,   p=<2e-16
## Wald test            = 218.1  on 8 df,   p=<2e-16
## Score (logrank) test = 279.3  on 8 df,   p=<2e-16
```

Next we will use the both versions of `coxed()` to examine expected durations and marginal changes in duration.

3 Using the NPSF method within the `coxed()` function

The first version of `coxed()` is the non-parametric step function (NPSF) approach. To use this version, specify `model="npsf"` in the call to `coxed()`. By default, quantities are estimated without standard errors, but to estimate SEs and confidence intervals specify `bootstrap=TRUE`.

3.1 Without standard errors

To see predicted durations from the Cox model, place the Cox model output as the first argument of `coxed()`:

```
ed1 <- coxed(mv.cox, method="npsf")
```

There are a number of uses of the `coxed()` output. First, the predicted durations for each individual observation are stored in the `exp.dur` attribute:

```
head(ed1$exp.dur)
```

```
##     exp.dur
## 1 34.620664
## 2 31.639975
## 3 39.509093
## 4 14.717544
## 5  2.473799
## 6 47.347670
```

The `summary()` function, when applied to `coxed`, reports either the mean or median duration in the estimation sample, depending on the option specified with `stat`:

```
summary(ed1, stat="mean")
```

```
##  mean
## 25.18
```

```
summary(ed1, stat="median")
```

```
## median
## 19.121
```

The predicted mean duration of government negotiations is 25.18 days, and the predicted median duration is 19.12 days.

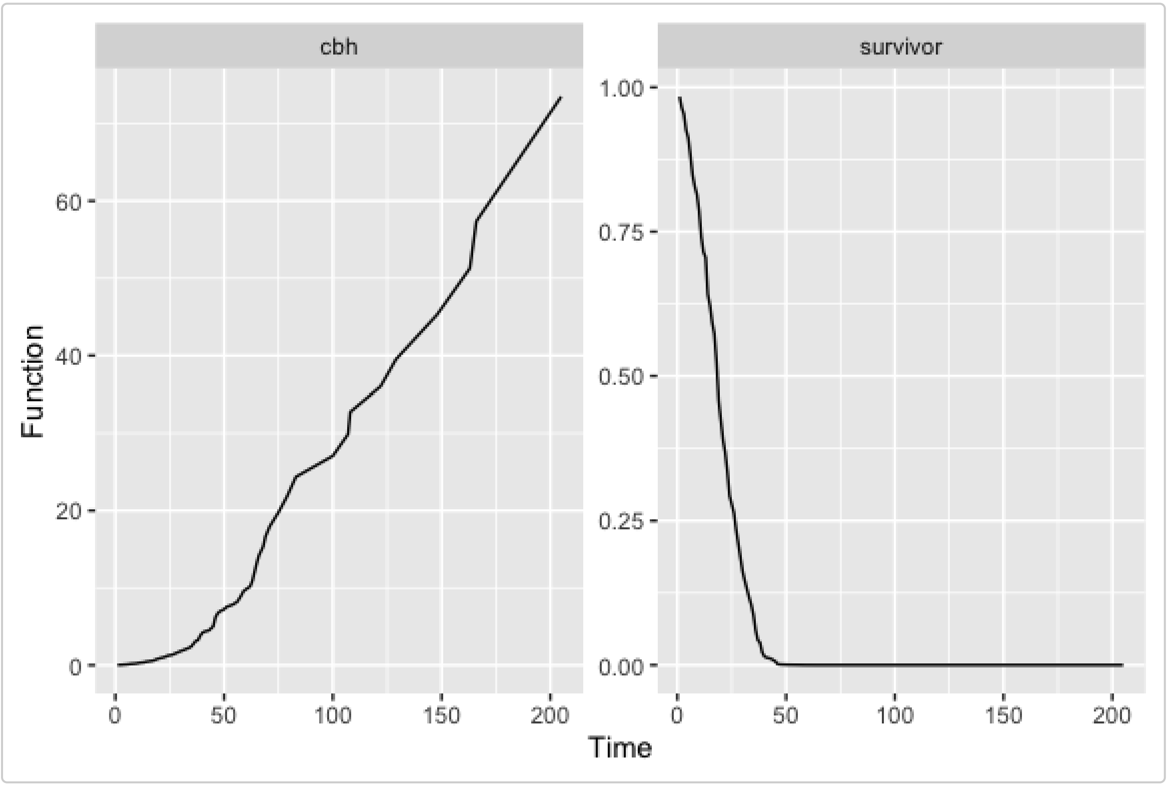
In addition to reporting the mean and median duration, the NPSF version of `coxed()` provides estimates of the cumulative baseline hazard function and the baseline survivor function in the data. These functions are stored as a data frame in the `baseline.functions` attribute.

```
head(ed1$baseline.functions)
```

```
##   time        cbh  survivor
## 1    1 0.01631230 0.9838200
## 2    2 0.03587341 0.9647624
## 3    3 0.04746259 0.9536462
## 4    4 0.07651484 0.9263392
## 5    5 0.09169880 0.9123799
## 6    6 0.12573075 0.8818522
```

We can plot these baseline functions with `ggplot()`:

```
baseline <- gather(ed1$baseline.functions, cbh, survivor, key="survivefunction", value="value")
ggplot(baseline, aes(x=time, y=value)) +
     geom_line() +
     xlab("Time") +
     ylab("Function") +
     facet_wrap( ~ survivefunction, scales = "free")
```



3.2 With (bootstrapped) standard errors

We can calculate standard errors and confidence intervals for any of these quantities with the `bootstrap=TRUE` option. By default the bootstrapping procedure uses 200 iterations (to set this value to a different number, use the `B` argument). Here we use 30 iterations simply to ease the computational burden of compiling this vignette. For more reliable results, set `B` to a higher value:

```
ed1 <- coxed(mv.cox, method="npsf", bootstrap = TRUE, B=30)
```

Now every predicted duration has a standard error and a 95% confidence interval.

```
head(ed1$exp.dur)
```

```
##     exp.dur bootstrap.se        lb        ub
## 1 34.620664    1.1286579 32.408535 36.832793
## 2 31.639975    1.0342026 29.612975 33.666975
## 3 39.509093    1.3384102 36.885857 42.132328
## 4 14.717544    0.6059843 13.529836 15.905251
## 5  2.473799    0.3026913  1.880535  3.067063
## 6 47.347670    2.0741821 43.282348 51.412992
```

The mean and median also have standard errors and confidence intervals.

```
summary(ed1, stat="mean")
```

```
##   mean bootstrap.se     lb     ub
##  25.18        0.957 23.304 27.055
```

```
summary(ed1, stat="median")
```

```
##  median bootstrap.se     lb     ub
##  19.121        0.708 17.733 20.508
```

To change the confidence interval to a different level, use the `level` argument:

```
ed1 <- coxed(mv.cox, method="npsf", bootstrap = TRUE, B=30, level=.8)
summary(ed1, stat="mean")
```

```
##   mean bootstrap.se     lb     ub
##  25.18        1.357 23.441 26.919
```

```
summary(ed1, stat="median")
```

```
##  median bootstrap.se   lb     ub
##  19.121        0.718 18.2 20.041
```

There are different methods for calculating a bootstrapped confidence interval. The default method used by `coxed()` (setting the argument `confidence="studentized"`) adds and subtracts `qnorm(level - (1 - level)/2)` times the bootstrapped standard error to the point estimate. The alternative approach is to take the `(1-level)/2` and `level + (1-level)/2` quantiles of the bootstrapped draws, which can be done by specifying `confidence="empirical"` (we recommend a higher number of bootstrap iterations for empirical confidence intervals):

```
ed1 <- coxed(mv.cox, method="npsf", bootstrap = TRUE, B=30, confidence="empirical")
summary(ed1, stat="mean")
```

```
##   mean bootstrap.se     lb     ub
##  25.18        1.305 22.615 26.996
```

```
summary(ed1, stat="median")
```

```
##  median bootstrap.se     lb    ub
##  19.121        0.687 17.693 20.05
```

3.3 Out of sample prediction

`coxed()` can be used to provide duration predictions for observations outside of the estimation sample. Suppose that we observe three new cases and place them inside a new data frame:

```
new.coalitions <- data.frame(postel = c(1,0,0),
                             prevdef = c(1,0,1),
                             cont = c(1,0,0),
                             ident = c(1,3,2),
                             rgovm = c(0.81, 0.62, 1.18),
                             pgovno = c(2,3,4),
                             tpgovno = c(3.58, 5.17, 10.2),
                             minority = c(0,0,1))
new.coalitions
```

```
##   postel prevdef cont ident rgovm pgovno tpgovno minority
## 1      1       1    1     1  0.81      2    3.58        0
## 2      0       0    0     3  0.62      3    5.17        0
## 3      0       1    0     2  1.18      4   10.20        1
```

To forecast durations for these cases along with standard errors and confidence intervals, we use the `coxed()` function and place `new.coalitions` into the `newdata` argument:

```
forecast <- coxed(mv.cox, newdata=new.coalitions, method="npsf", bootstrap=TRUE, B=30)
forecast$exp.dur
```

```
##     exp.dur bootstrap.se       lb        ub
## 1  6.758058     2.107153 2.628113 10.888003
## 2  4.805083     1.788725 1.299247  8.310919
## 3 16.489185     4.171693 8.312816 24.665553
```

3.4 Marginal changes in expected duration

Here we use `coxed()` to provide answers to the two duration-based questions we posed in the introduction. First consider “How much longer will negotiations take for an ideologically polarized coalition as compared to an ideologically homogeneous one?” To answer this question, we call `coxed()` and specify two new datasets, one in which `rgovm=0` indicating that all political parties in the governing coalition have the same ideological position, and one in which `rgovm=1.24`, indicating that the parties have very different ideological positions. We use `mutate()` from the `dplyr` library to quickly create new data frames in which `rgovm` equals 0 or 1.24 for all cases, and set these two data frames as `newdata` and `newdata2` inside `coxed()`.

```
me <- coxed(mv.cox, method = "gam", bootstrap = TRUE, B = 30,
            newdata = dplyr::mutate(martinvanberg, rgovm = 0),
            newdata2 = dplyr::mutate(martinvanberg, rgovm = 1.24))
```

```
## Warning in rank.predict(x = exp.xb2, v = exp.xb, warn = warn): New data
## contain 2 observations with linear predictors less than or equal to all linear
## predictors in the estimation sample. These observations will all have the same
## predicted duration

## Warning in rank.predict(x = exp.xb2, v = exp.xb, warn = warn): New data
## contain 2 observations with linear predictors less than or equal to all linear
## predictors in the estimation sample. These observations will all have the same
## predicted duration

## Warning in rank.predict(x = exp.xb2, v = exp.xb, warn = warn): New data
## contain 2 observations with linear predictors less than or equal to all linear
## predictors in the estimation sample. These observations will all have the same
## predicted duration
```

`coxed()` calculates expected durations for all cases under each new data frame and subtracts the durations for each case. As an overall result, we can see either the mean or the median of these differences.

```
summary(me, stat="mean")
```

```
##              mean bootstrap.se     lb     ub
## newdata2   29.574        3.412 22.885 36.262
## newdata    25.926        2.490 21.045 30.806
## difference  3.648        1.963 -0.199  7.495
```

```
summary(me, stat="median")
```

```
##            median bootstrap.se     lb     ub
## newdata2   22.642        2.858 17.041 28.243
## newdata    19.909        2.240 15.519 24.299
## difference  3.235        1.534  0.228  6.242
```

A coalition in which the parties have ideological differences so that `rgovm=1.24` will take 3.08 more days on average (with a median of 2.6 days) to conclude negotiations than a coalition in which all parties have the same position.

Next we consider “How much longer will negotiations take for a multiparty coalition government than for a single-party government?” In this case we compare coalitions with one party to coalitions with 6 parties by setting the `pgovno` variable to 1 and 6 and setting these two data frames as the `newdata` and `newdata2` arguments of `coxed()`:

```
me <- coxed(mv.cox, method="npsf", bootstrap = TRUE, B=30,
            newdata = dplyr::mutate(martinvanberg, pgovno=1),
            newdata2 = dplyr::mutate(martinvanberg, pgovno=6))
summary(me, stat="mean")
```

```
##               mean bootstrap.se      lb      ub
## newdata2     5.801        0.914   4.011   7.592
## newdata     56.298        8.527  39.585  73.011
## difference -50.496        8.531 -67.217 -33.776
```

```
summary(me, stat="median")
```

```
##             median bootstrap.se      lb      ub
## newdata2     0.008        0.006  -0.004   0.019
## newdata     28.508        4.069  20.533  36.483
## difference -28.500        2.942 -34.266 -22.734
```

A coalition of 6 parties will take 50.5 more days on average (with a median of 28.5 days) to conclude negotiations than a coalition with one party.

4 Using the GAM method within the `coxed()` function

We can use the GAM method to for all of the same uses for which we used the NPSF method above, except for estimating the baseline functions. We can however view and plot the output from the GAM model that maps predicted ranks to duration. While the `bootstrap=TRUE` argument works when `method="gam"`, these functions take somewhat longer to run. We therefore run the following examples without bootstrapping.

As before, to see predicted durations from the Cox model, place the Cox model output as the first argument of `coxed()`. The predicted durations for each individual observation are stored in the `exp.dur` attribute,

```
ed2 <- coxed(mv.cox, method="gam")
head(ed2$exp.dur)
```

```
##     exp.dur
## 1 48.945416
## 2 41.993432
## 3 55.420164
## 4 15.734190
## 5  1.545759
## 6 64.433227
```

and `summary()` reports either the mean or median expected duration:

```
summary(ed2, stat="mean")
```

```
##   mean
## 28.034
```

```
summary(ed2, stat="median")
```

```
## median
## 21.131
```

4.1 Out of sample prediction

The GAM method can also forecast durations for new data along with standard errors and confidence intervals. Here we use the `coxed()` function with `method="gam"` and place the `new.coalitions` we created above into the `newdata` argument:

```
forecast <- coxed(mv.cox, newdata=new.coalitions, method="gam")
forecast$exp.dur
```

```
##     exp.dur
## 1  6.778237
## 2  3.328481
## 3 18.392858
```

4.2 Marginal changes in expected duration

Here we again calculate the two marginal effects to better understand the substantive meaning of the Cox model. This time we employ the GAM method instead of the NPSF method. The GAM method may provide a warning that some observations have linear predictors that are greater than or less than all of the observed cases in the estimation sample. Some observations falling outside the range of the original linear predictors is to be expected when applying new data, but if it happens with too many of the new observations NPSF may be a better option for estimating these quantities.

```
me <- coxed(mv.cox, method="gam",
            newdata = dplyr::mutate(martinvanberg, rgovm=0),
            newdata2 = dplyr::mutate(martinvanberg, rgovm=1.24))
```

```
## Warning in rank.predict(x = exp.xb2, v = exp.xb, warn = warn): New data
## contain 2 observations with linear predictors less than or equal to all linear
## predictors in the estimation sample. These observations will all have the same
## predicted duration
```

```
summary(me, stat="mean")
```

```
##   newdata2    newdata difference
##     25.926     29.574      3.648
```

```
summary(me, stat="median")
```

```
##   newdata2    newdata difference
##     19.909     22.642      3.235
```

```
me <- coxed(mv.cox, method="gam",
            newdata = dplyr::mutate(martinvanberg, pgovno=1),
            newdata2 = dplyr::mutate(martinvanberg, pgovno=6))
```

```
## Warning in rank.predict(x = exp.xb2, v = exp.xb, warn = warn): New data
## contain 20 observations with linear predictors less than or equal to all linear
## predictors in the estimation sample. These observations will all have the same
## predicted duration
```

```
summary(me, stat="mean")
```

```
##   newdata2    newdata difference
##     46.795      7.938    -38.857
```

```
summary(me, stat="median")
```

```
##   newdata2    newdata difference
##     35.167      1.858    -31.349
```

4.3 Plotting the GAM fit

The data used by `coxed()` to map rankings to durations are stored in the `gam.data` attribute, and the output from the GAM is stored in `gam.model`:

```
summary(ed2$gam.data)
```

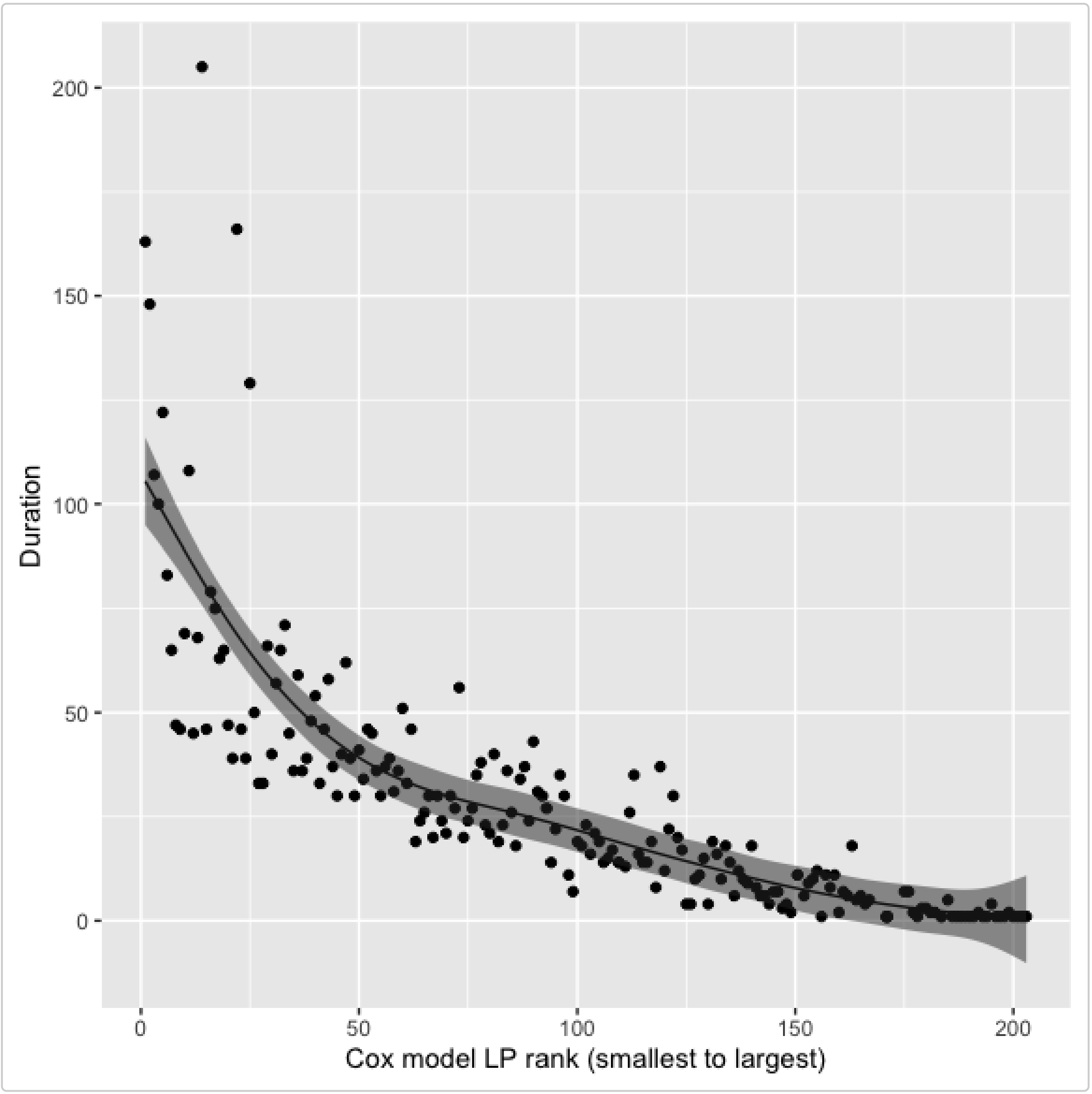
```
##        y              failed     rank.xb          rank.y     
##  Min.   :  1.00   Min.   :1   Min.   :  1.0   Min.   : 13.5  
##  1st Qu.:  6.00   1st Qu.:1   1st Qu.: 51.5   1st Qu.: 49.5  
##  Median : 19.00   Median :1   Median :102.0   Median : 99.5  
##  Mean   : 28.03   Mean   :1   Mean   :102.0   Mean   :102.0  
##  3rd Qu.: 37.50   3rd Qu.:1   3rd Qu.:152.5   3rd Qu.:151.8  
##  Max.   :205.00   Max.   :1   Max.   :203.0   Max.   :203.0  
##     gam_fit            gam_fit_se     gam_fit_95lb       gam_fit_95ub   
##  Min.   : 0.3225   Min.   :2.636   Min.   :-10.196   Min.   :  7.484  
##  1st Qu.: 7.3071   1st Qu.:2.702   1st Qu.:  1.852   1st Qu.: 12.763  
##  Median : 21.1306   Median :2.757   Median : 15.836   Median : 26.425  
##  Mean   : 28.0345   Mean   :2.923   Mean   : 22.305   Mean   : 33.764
##  3rd Qu.: 38.2325   3rd Qu.:2.821   3rd Qu.: 32.931   3rd Qu.: 43.534
##  Max.   :105.4596   Max.   :5.385   Max.   : 94.905   Max.   :116.014
```

```
summary(ed2$gam.model)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## y ~ s(rank.xb, bs = "cr", k = k)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   28.034      1.198    23.4   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##              edf Ref.df     F p-value
## s(rank.xb) 5.141  6.233 77.28  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.704   Deviance explained = 71.2%
## GCV = 300.51  Scale est. = 291.42    n = 203
```

The `gam.data` can be used to visualize the fit of the GAM:

```
ggplot(ed2$gam.data, aes(x=rank.xb, y=y)) +
     geom_point() +
     geom_line(aes(x=rank.xb, y=gam_fit)) +
     geom_ribbon(aes(ymin=gam_fit_95lb, ymax=gam_fit_95ub), alpha=.5) +
     xlab("Cox model LP rank (smallest to largest)") +
     ylab("Duration")
```



5 Comparing the predicted durations to the observed durations and to each other

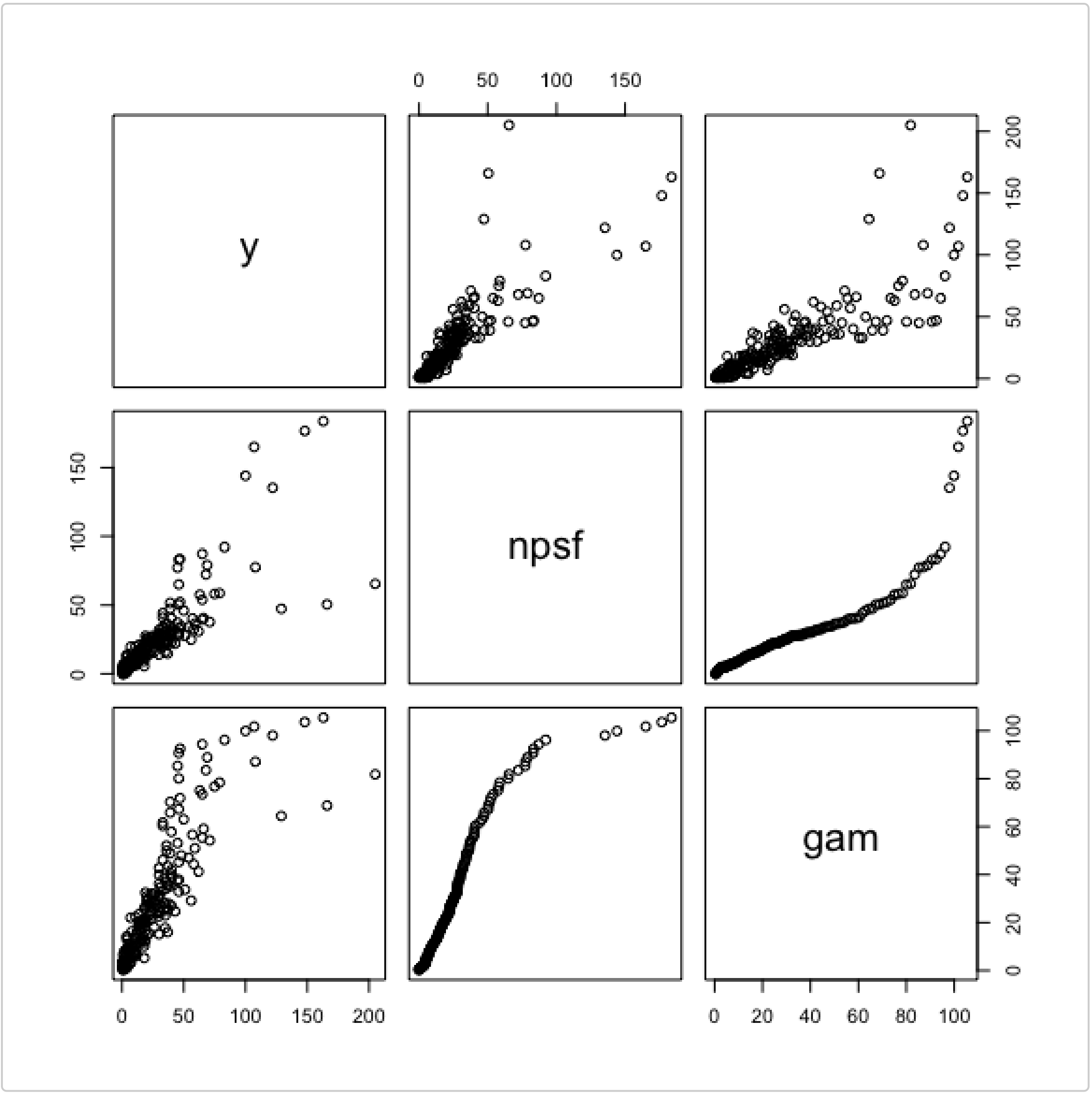
Given that `coxed()` contains two alternative methods for generating expected durations, it is possible to compare the estimates. Both correlate positively the observed durations, and the GAM and NPSF durations correlate even more strongly with each other.

```
tester <- data.frame(y=martinvanberg$formdur, npsf=ed1$exp.dur$exp.dur, gam=ed2$exp.dur$exp.dur)
cor(tester)
```

```
##              y      npsf       gam
## y    1.0000000 0.8240072 0.8436852
## npsf 0.8240072 1.0000000 0.9122332
## gam  0.8436852 0.9122332 1.0000000
```

Scatterplots visualize these correlations:

```
pairs(tester)
```



6 Expected durations and marginal changes in duration with time-varying covariates

To illustrate the use of `coxed()` with time-varying covariates, we use another working example. To set up this example, we quote from the online appendix to Kropko and Harden (2018):

> Box-Steffensmeier (1996) examines whether U.S. House incumbents' ability to raise campaign funds can effectively deter quality challengers from entering the race. The theoretical expectation is that as incumbents raise more money, challengers further delay their decision to run for the incumbent's seat. She employs data on 397 House races in the 1989–1990 election cycle to test this hypothesis. The dependent variable in this analysis is the number of weeks after January 1, 1989 when a challenger entered the race. Races in which no challenger entered are coded as the number of weeks after January 1 when the state's primary filing deadline occurred, and are treated as censored. The key independent variable is the incumbent's War chest, or the amount of money in millions of dollars that the incumbent has in reserve at a given time. Importantly, this measure updates over the course of five Federal Election Commission (FEC) reporting periods, so it is a time-varying covariate (TVC). The theory predicts a negative coefficient on this variable, which would

indicate that as the incumbent raises more money, the hazard of challenger entry declines (and the time until entry increases).

Box-Steffensmeier’s model is replicated below. Note that the `Surv()` function which sets up the dependent variable has two time arguments, representing the start and end of discrete intervals, which allows a covariate to take on different values across different intervals for the same observation.

```
bs.surv <- Surv(time = boxsteffensmeier$start, time2 = boxsteffensmeier$te, event =
  boxsteffensmeier$cut_hi)
bs.cox <- coxph(bs.surv ~ ec + dem + south + iv, data = boxsteffensmeier, method = "breslow")
summary(bs.cox)
```

```
## Call:
## coxph(formula = bs.surv ~ ec + dem + south + iv, data = boxsteffensmeier,
##     method = "breslow")
##
##   n= 1376, number of events= 40
##
##              coef  exp(coef)   se(coef)      z Pr(>|z|)
## ec    -3.0305806  0.0482876  1.3961518 -2.171    0.030 *
## dem    0.2124840  1.2367464  0.3266545  0.650    0.515
## south -0.4266388  0.6526992  0.4240698 -1.006    0.314
## iv    -7.2790246  0.0006899  1.6996244 -4.283 1.85e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##       exp(coef) exp(-coef) lower .95 upper .95
## ec    0.0482876    20.7093 3.129e-03    0.7451
## dem   1.2367464     0.8086 6.520e-01    2.3460
## south 0.6526992     1.5321 2.843e-01    1.4986
## iv    0.0006899  1449.5734 2.466e-05    0.0193
##
## Concordance= 0.773  (se = 0.044 )
## Likelihood ratio test= 35.04  on 4 df,   p=5e-07
## Wald test            = 25.52  on 4 df,   p=4e-05
## Score (logrank) test = 26.9  on 4 df,   p=2e-05
```

The `coxed()` function automatically detects whether time-varying covariates are used in the model and it takes steps to account for this structure in predicting expected durations and in estimating marginal effects. The only additional step that the user needs to take is to specify the ID variable in the `id` argument, so that the function knows which intervals refer to which observations. The `id` variable must be the ID from the data that was used to estimate the Cox PH model, and NOT the ID variable in any new data frame.

```
ed1 <- coxed(bs.cox, method="npsf", id=boxsteffensmeier$caseid)
summary(ed1, stat="mean")
```

```
##   mean
## 85.128
```

Here we look directly at the effect of the war chest on the length of time until a high quality challenger enters the race. We compare the 25th and 75th percentiles in war chest variable:

```
me <- coxed(bs.cox, method="npsf",
            newdata = mutate(boxsteffensmeier, ec=quantile(ec, .25)),
            newdata2 = mutate(boxsteffensmeier, ec=quantile(ec, .75)),
            id=boxsteffensmeier$caseid)
summary(me, stat="mean")
```

```
##   newdata2    newdata difference
##     84.047     86.520      2.473
```

```
summary(me, stat="median")
```

```
##   newdata2    newdata difference
##     84.873     87.086      2.214
```

An incumbent whose war chest is at the 75th percentile in the data delays the entry of a high quality challenger by 2.7 weeks (or 2.4 weeks when evaluated at the medians), on average, compared to an incumbent whose war chest is at the 25th percentile.

7 References

- Box-Steffensmeier, J. M. (1996) “A Dynamic Analysis of The Role of War Chests in Campaign Strategy.” *American Journal of Political Science* **40**: 352-371.
- Kropko, J. and Harden, J. J. (2018) “Beyond the Hazard Ratio: Generating Expected Durations from the Cox Proportional Hazards Model.” *British Journal of Political Science* https://doi.org/10.1017/S000712341700045X
- Martin, L. W and Vanberg, G. (2003) “Wasting Time? The Impact of Ideology and Size on Delay in Coalition Formation.” *British Journal of Political Science* **33** 323-344 https://doi.org/10.1017/S0007123403000140