A Bootstrap Validated Parsimonious Cox PH Regression Model Predicting AIDS Survival

Madison Hobbs & Lathan Liou

May 1, 2019

Introduction

The AIDS epidemic exploded in the early 80s, and the scientific community raced to find curative treatments. One such research group, Hammer et. al., sought to perform a study using a double-blind placebo controlled trial to compare the three-drug regimen of indinavir (IDV), zidovudine (ZDV) or stavudine (d4T) and lamivudine (3TC) with the two-drug regimen of ZDV or d4T and 3TC in HIV-infected patients. Specifically, they were interested in testing the effects of IDV, or as it's more commonly known, Crixivan. The discoverer of Crixivan, Paul Reider (who lectured at Pomona last year as a Robbins Chemistry Fellow!), had high hopes for this drug given its unique 3-D binding capacity to inhibit specific proteins in HIV.

Having obtained the data from the study, our project seeks to understand the factors associated with time of survival until an AIDS defining event or death. We first seek to understand the distributions of our time-to-event variables and the other explanatory variables through an exploratory data analysis. Next, we aim to identify the most important predictors by using a Cox Proportional Hazards model and an XGBoost model. Lastly, we seek to validate our model by a bootstrap analysis. We also have hidden most of our code in this report for readability purposes; however, you can find our full code on our GitHub: https://github.com/latlio/SurvivalAnalysis.

Our analysis illuminates the importance of lower CD4 count and the IDV treatment as predictors associated with lower risk of AIDS diagnosis or death.

Exploratory Data Analysis

Exploratory Data Analysis (EDA) is, and should be, the *first* step when working with data. Our primary objective is to visualize the distributions of our variables to gain a clearer understanding of how they vary.

A Note About Treatments

According to the variable information table, we note that `txgrp` could have four levels (1: ZDV + 3TC, 2: ZDV + 3TC + IDV, 3: d4T + 3TC, and 4: d4T + 3TC + IDV). However, this dataset contains only two levels of `txgrp` (1: ZDV + 3TC, 2: ZDV + 3TC + IDV).

In fact, since the variable `tx` is supposed to indicate whether the treatment contained IDV, we might assume that `txgrp` and `tx` are redundant information in this particular dataset and that a 1 in `txgrp` is equivalent to a 0 in `tx` while a 2 in `txgrp` is equivalent to a 1 in `tx`. We confirm this hunch below.

The following code says: create a new dataframe by taking all the rows in `data` where `txgrp` is 1 and `tx` is 0 *or* `txgrp` is 2 and `tx` is 1. Now, make sure that new dataframe is identical to the original data frame, and return `TRUE` if this is indeed the case.

```
# Is it true that for every entry in `aids`
all(
  (aids %>%
```

```
    filter(((txgrp == 1 && tx == 0) || (txgrp == 2 && tx == 1))))
  == aids) == TRUE
```

```
## [1] TRUE
```

Exploring Correlation

From an initial pair-wise correlation plot between our explanatory variables in our data (not shown here), we noticed that `cd4` and `strat2` have a correlation coefficient of 0.74, which indicates moderate to strong correlation. This made us realize that `strat2` is the indicator variable for the continuous variable, `cd4`. Additionally, we confirmed that `tx` and `txgrp` are identical because they were perfectly correlated. Lastly, we would like to note that `sex`, `ivdrug`, and `hemophil` are highly unbalanced variables, meaning that one level of the variables is disproportionately represented relative to the other level(s).

A Note about Censored vs. Non-Censored

It's worth noting that there are, in fact, two censored time-to-event variables. The primary variable of interest is `time` which is time in days to AIDs diagnosis or death, and this is informed by `censor`, which is 1 (true) if an individual was either diagnosed with AIDS *or* died during the course of the study and 0 otherwise. The other censored variable is `time_d` which is the time in days to death alone, governed by `censor_d` which is 1 if the person died during the study and 0 if not.

Since the primary variable of interest is time to AIDs diagnosis or death, we examine the complete (non-censored) individuals - those who were either diagnosed with AIDS *or* who died over the course of the study. The only caveat is that there are only 69 such individuals out of a study of 851 - most of the participants did not die or get diagnosed before the study's end.

Among those with complete times, we notice from the side-by-side histograms below that both the control and IDV groups are skewed right. This makes sense - for complete observations, it's probably less common for people to last a long time without being diagnosed or dying. The distributions between the control and IDV groups don't look that different, however, especially given the tiny sample sizes.

Figure 1. Comparing the Distribution of Complete Time to Event Observations Between Control and Treatment
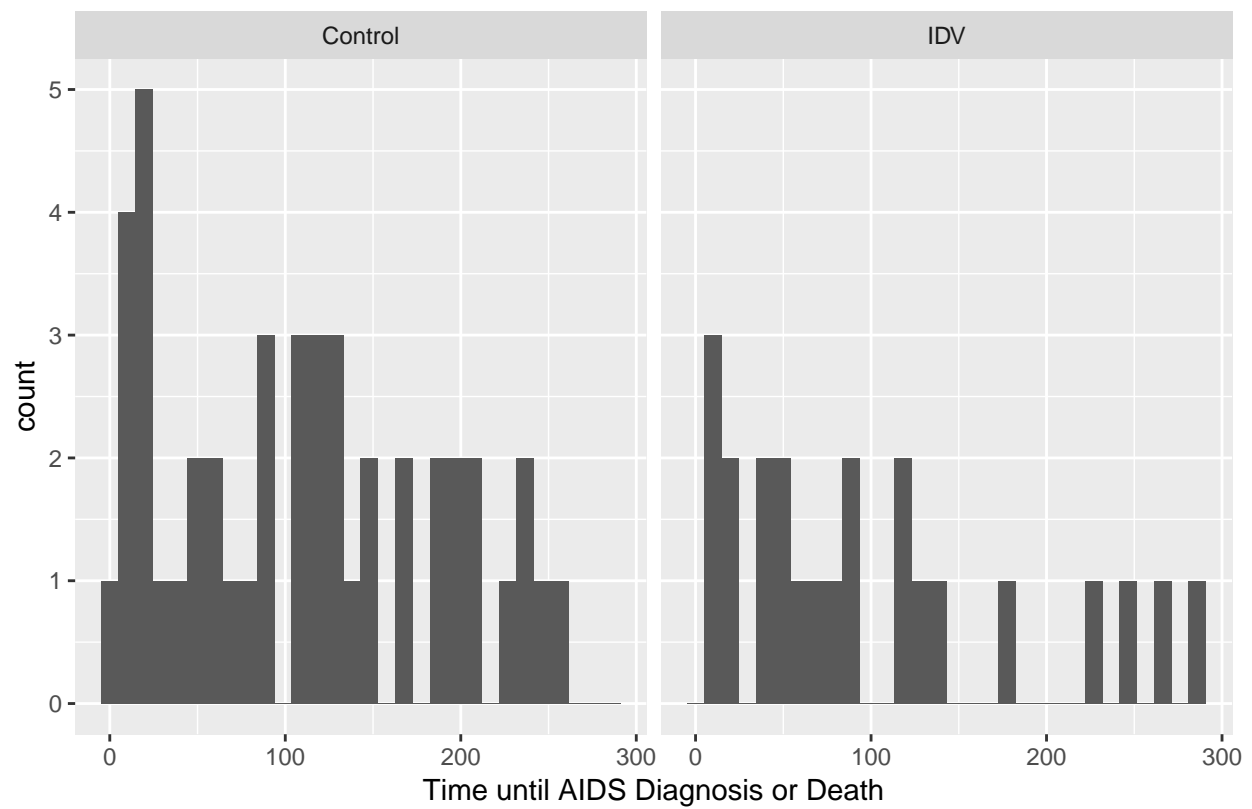
When looking at the censored (incomplete) times for diagnosis/death, both control and IDV groups are skewed left in the opposite direction.
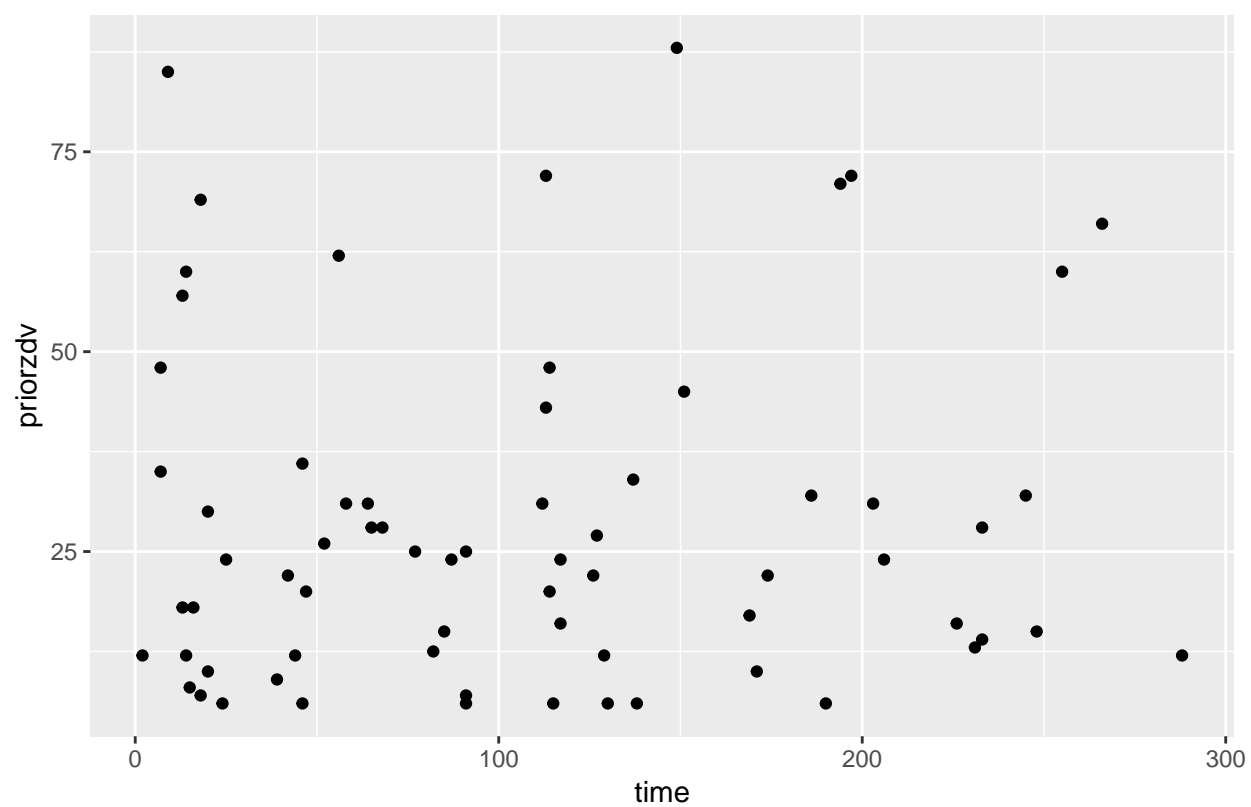
Figure 2. Comparing the Distribution of Incomplete Time to Event Observations Between Control and Treatme
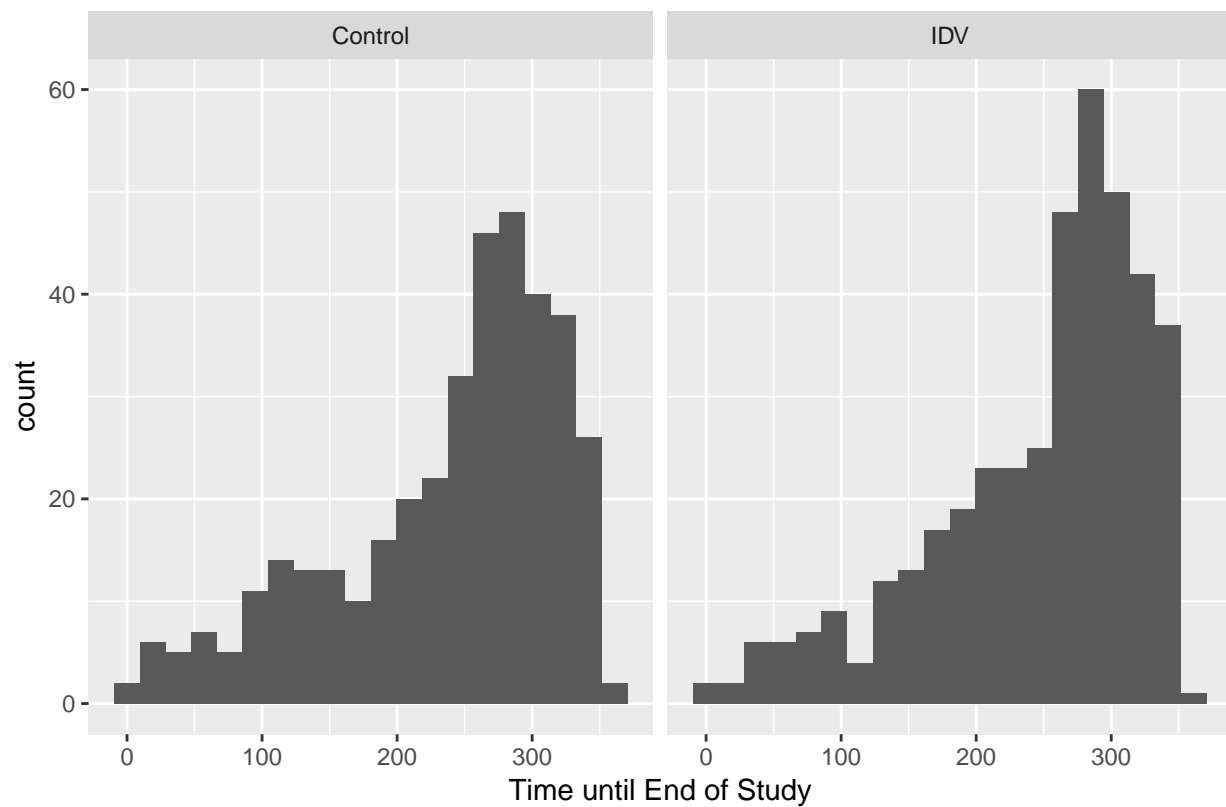
## Prior ZDV on Complete Observations

We were also curious about the relationship between time to diagnosis/death and number of months of prior ZDV use for non-censored participants, since ZDV is one of the drugs in both the treatment and control regimen. Interestingly, there appeared to be no relationship whatsoever, as evidenced by the following scatterplot:

Figure 3. Scatterplot of Number of Months of Prior ZDV Treatment vs. Time to Death/Diagnosis

Figure 4. Survival Curve of Time to Event Variable

Strata   ─╂─   All
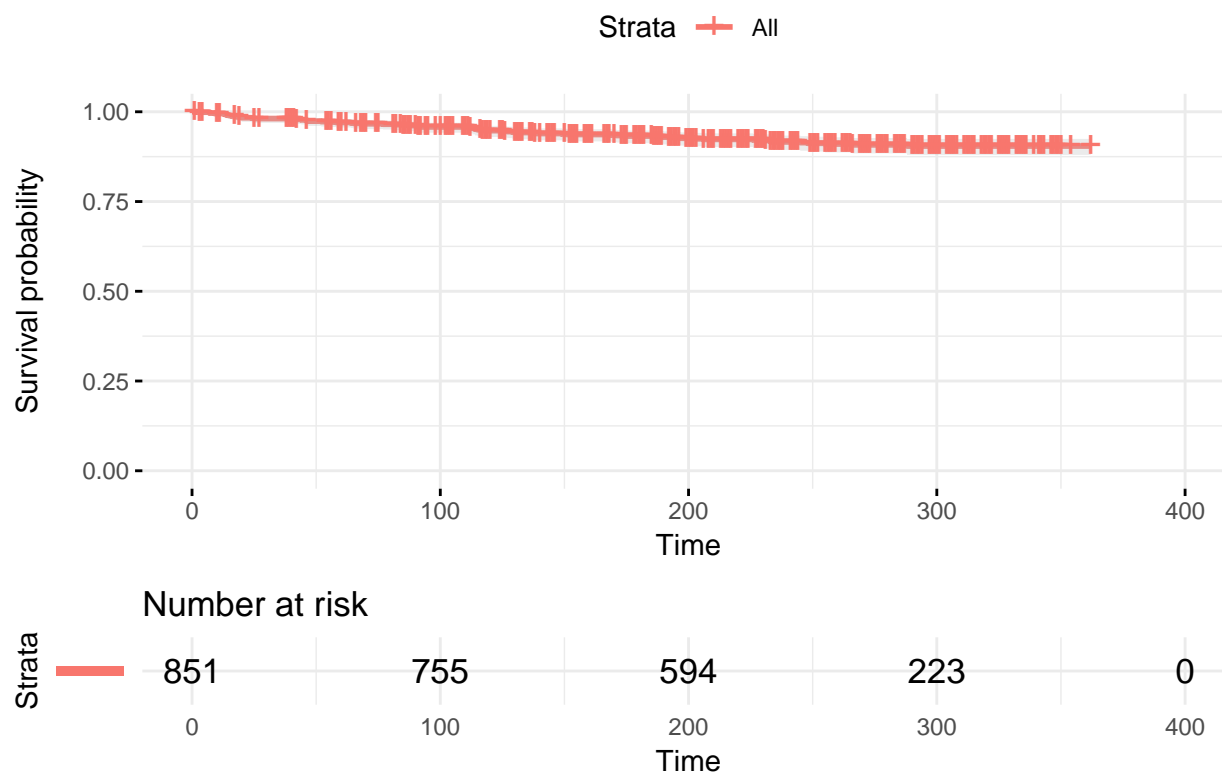
We also fit a survival curve of just the time to death variable. We observe that the overall survival probability of our sample remains relatively high over time and that the last observation is censored. Because treatment is the clinical variable of interest, we next want to see how the survival curves differ between the two treatment groups.

Figure 5. Survival Curves Comparing Control to Treatment of Time to Ev...
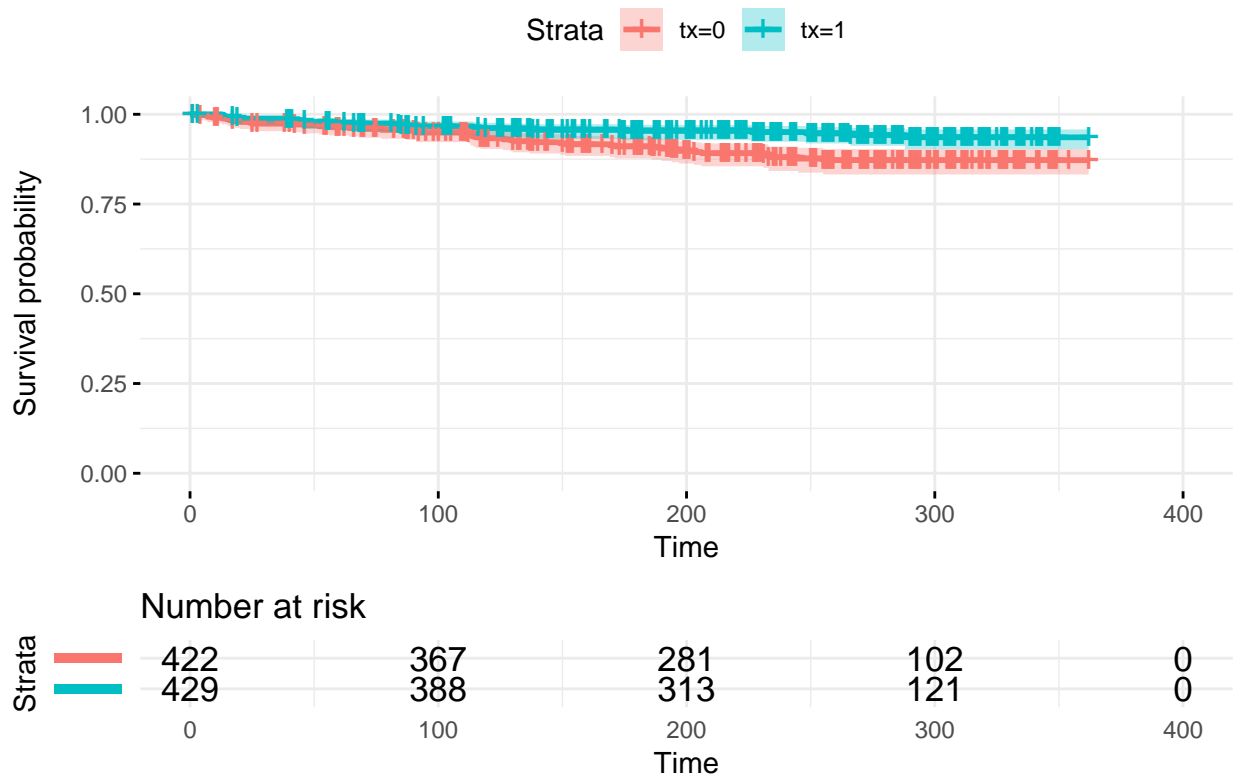
```
#perform log-rank test
survdiff(Surv(time, censor) ~ tx, data = data, rho = 1)
```

```
## Call:
## survdiff(formula = Surv(time, censor) ~ tx, data = data, rho = 1)
##
##         N Observed Expected (O-E)^2/E (O-E)^2/V
## tx=0 422     44.0     31.9      4.57      9.24
## tx=1 429     22.1     34.2      4.28      9.24
##
##  Chisq= 9.2  on 1 degrees of freedom, p= 0.002
```

We see that the treatment group for which IDV was also administered has a higher survival probability over time compared to the control group. Performing the log-rank test to test our null hypothesis of whether $S_0(t) = S_1(t)$ for all t results in a $\chi^2_1 = 9.2$ with a p-value of 0.002. We thus reject the null hypothesis and conclude that there is evidence that supports that the survival probabilities are significantly different between the treatment groups over some time intervals.

Overall, from our Exploratory Data Analysis, we have several key takeaways that will guide us in our modeling process:

- We will use 'tx' and not 'txgrp' in our model since they are identical in this particular dataset

- We will use 'cd4' and not 'strat2' in our model since 'strat2' is the indicator variable for cd4.

- 'Priorzdv' may not be an important predictor in our later models, e.g. our Cox PH model, whereas Treatment might be.

Cox Proportional Hazards Model

Choosing the Number of Parameters

Our goal is to develop a multivariable survival model for time until death (or diagnosis). In particular, our objective is to build the best predictive model, i.e. we want the highest accuracy on new data. There are 69 deaths (or diagnoses) among 782 patients. The first thing we want to assess is a full additive model. Thus, categorical predictors were expanded using dummy variables. We chose not to include `txgrp` and `strat2` because they were derived (and thus highly correlated with) from other predictor variables.

First, we make sure that the technical condition for proportional hazards is met with the hypothesis test below.

```
cox.zph(coxph(Surv(time,censor) ~ tx + sex + raceth + ivdrug + hemophil + karnof + cd4 + priorzdv + age
```

```
##                rho    chisq      p
## tx1        -0.12102 1.07e+00 0.3009
## sex2       -0.15899 1.86e+00 0.1726
## raceth2     0.19389 2.84e+00 0.0917
## raceth3     0.12322 1.04e+00 0.3086
## raceth4    -0.07600 4.48e-01 0.5032
## raceth5     0.15584 1.34e+00 0.2472
## ivdrug2    -0.18070 7.26e-08 0.9998
## ivdrug3    -0.07277 3.54e-01 0.5516
## hemophil1   0.05969 2.08e-01 0.6486
## karnof80    0.00843 5.08e-03 0.9432
## karnof90    0.02646 5.60e-02 0.8129
## karnof100  -0.06425 2.88e-01 0.5912
## cd4         0.12594 1.02e+00 0.3120
## priorzdv    0.06080 1.78e-01 0.6732
## age         0.15026 1.68e+00 0.1950
## GLOBAL           NA 1.24e+01 0.6448
```

Since no p-values are significant at the $\alpha = 0.05$ level, we conclude that there are no violations of the proportional hazards assumption, so we continue building our CoxPH model. We first build a full additive model.

```
options(scipen = 999)
fit <- coxph(Surv(time, censor) ~ tx + sex + raceth + ivdrug + hemophil + karnof + cd4 + priorzdv + age
fit %>% tidy()
```

```
##         term      estimate      std.error     statistic       p.value
## 1        tx1  -0.692689185   0.259981564 -2.664378094 0.00771308167
## 2       sex2   0.433955810   0.330969192  1.311166782 0.18980142225
## 3    raceth2  -0.266199123   0.306770367 -0.867747189 0.38553274649
## 4    raceth3  -0.133104042   0.350590733 -0.379656474 0.70420043356
## 5    raceth4   0.880199385   0.740160204  1.189201176 0.23436051188
## 6    raceth5   0.159164019   1.062271128  0.149833705 0.88089581703
## 7    ivdrug2 -13.662745199 2217.947151763 -0.006160086 0.99508499357
## 8    ivdrug3  -0.518173596   0.371149088 -1.396133284 0.16267436244
## 9   hemophil1   0.505671649   0.620652558  0.814741907 0.41522005977
## 10   karnof80  -0.690301943   0.429072156 -1.608824839 0.10765464789
## 11   karnof90  -1.349833322   0.428707075 -3.148614520 0.00164046432
## 12  karnof100  -1.810573691   0.475762899 -3.805621862 0.00014144835
## 13        cd4  -0.015164686   0.003190856 -4.752544063 0.00000200873
## 14   priorzdv  -0.002115753   0.004726267 -0.447658398 0.65439974999
```

```
## 15        age   0.025738322     0.013850413  1.858307191 0.06312540323
##         conf.low    conf.high
## 1  -1.202243688 -0.183134682
## 2  -0.214731886  1.082643506
## 3  -0.867457993  0.335059747
## 4  -0.820249252  0.554041168
## 5  -0.570487958  2.330886729
## 6  -1.922849135  2.241177172
## 7          -Inf          Inf
## 8  -1.245612442  0.209265250
## 9  -0.710785012  1.722128310
## 10 -1.531267916  0.150664030
## 11 -2.190083750 -0.509582895
## 12 -2.743051839 -0.878095543
## 13 -0.021418649 -0.008910722
## 14 -0.011379067  0.007147561
## 15 -0.001407989  0.052884633
```

The likelihood ratio χ^2 statistic is 91.05 with 21 d.f of our full additive model. After considering whether variables can be mutating into new variables based on our conventional knowledge and finding none, we decided to try shrinkage to reduce our dimensionality. Here, we're using a lasso penalty Cox PH regression model to select our most important features. In brief, lasso essentially imposes an absolute value threshold penalty on each of the β coefficients such that the variables that don't contribute much to explaining the response have their coefficients shrunk to zero.

```r
set.seed(47)
#initialize covariate matrix
x <- model.matrix(Surv(time, censor) ~ tx + sex + raceth + ivdrug + hemophil + karnof + cd4 + priorzdv
#cross validate lambda
cv.fit <- cv.glmnet(x, Surv(data$time, data$censor), family = "cox", maxit = 1000)

lassofit <- glmnet(x, Surv(data$time, data$censor), family = "cox", maxit = 1000)
#see which coefficients were kept
active.coefs <- predict(lassofit, type = 'coefficients', s = cv.fit$lambda.min)
```

Variable	Lasso-ed β
(Intercept)	0.00
tx1	-0.58
sex2	0.13
raceth2	-0.07
raceth3	0.00*
raceth4	0.64
raceth5	0.00*
ivdrug2	-0.06
ivdrug3	-0.24
hemophil1	0.04
karnof80	0.00*
karnof90	-0.62
karnof100	-0.98
cd4	-0.01
priorzdv	0.00*
age	0.01

We see that the dummy variable for Hispanic and American Indian, the dummy variable for a Karnofsky score of 80 and priorzdv were shrunk to 0. If we rerun our Cox PH model without priorzdv, which from our

EDA was found to not be highly correlated with time, and conduct a likelihood ratio test, we find that it's not needed in the model based on a result of a $\chi_1^2 = 0.2086$ and a p-value of 0.6479

```
fit2 <- coxph(Surv(time, censor) ~ tx + sex + raceth + ivdrug + hemophil + karnof + cd4 + age, data = d
anova(fit, fit2)

## Analysis of Deviance Table
##  Cox model: response is  Surv(time, censor)
##  Model 1: ~ tx + sex + raceth + ivdrug + hemophil + karnof + cd4 + priorzdv + age
##  Model 2: ~ tx + sex + raceth + ivdrug + hemophil + karnof + cd4 + age
##     loglik  Chisq Df P(>|Chi|)
## 1 -410.34
## 2 -410.45 0.2086  1    0.6479
```

We do think our model can be more parsimonious; however, so as to avoid overspecification, we look back at the Wald's p-values of the full additive model, and we see that treatment, karnof, cd4, and age (slightly above 0.05) are statistically significant. Before proceeding, we acknowledge that there is a fine line between trying not to overspecify our model and missing potential predictor variables that can contribute some explanatory power. We fit a model with only those 4 variables and conduct a likelihood ratio test between this model and the additive model without `priorzdv`.

```
fit3 <- coxph(Surv(time, censor) ~ tx + karnof + cd4 + age, data = data)
anova(fit2, fit3)

## Analysis of Deviance Table
##  Cox model: response is  Surv(time, censor)
##  Model 1: ~ tx + sex + raceth + ivdrug + hemophil + karnof + cd4 + age
##  Model 2: ~ tx + karnof + cd4 + age
##     loglik  Chisq Df P(>|Chi|)
## 1 -410.45
## 2 -413.81 6.7234  8    0.5667
```

With a $\chi_8^2 = 6.72$ and a p-value of 0.5667, we conclude that none of the other variables in the additive model were needed.

Because age had a borderline p-value, we tried removing it from the model and seeing whether it was important or not.

```
fit4 <- coxph(Surv(time, censor) ~ tx + karnof + cd4, data = data)
anova(fit3, fit4)

## Analysis of Deviance Table
##  Cox model: response is  Surv(time, censor)
##  Model 1: ~ tx + karnof + cd4 + age
##  Model 2: ~ tx + karnof + cd4
##     loglik  Chisq Df P(>|Chi|)
## 1 -413.81
## 2 -414.92 2.2308  1    0.1353
```

It turns out, with a $\chi_1^2 = 2.23$ and a p-value of 0.1353, that age is not needed in the model.

One nagging thought is that marginal variables might have *some* real predictive value even if it's slight. To that end, let's test whether interactions are significant or not. Specifically, because we have reason to believe that there my be interacting effects with treatment group (the clinical variable of interest), we decided to interact treatment with our categorical covariates along with adjusting for cd4, priorzdv, and age.

```
fit.int <- coxph(Surv(time, censor) ~ tx*sex + tx*raceth + tx*ivdrug + tx*hemophil + tx*karnof + cd4 + p
anova(fit, fit.int)

## Analysis of Deviance Table
```

```
##  Cox model: response is  Surv(time, censor)
##  Model 1: ~ tx + sex + raceth + ivdrug + hemophil + karnof + cd4 + priorzdv + age
##  Model 2: ~ tx * sex + tx * raceth + tx * ivdrug + tx * hemophil + tx * karnof + cd4 + priorzdv + ag
##    loglik  Chisq Df P(>|Chi|)
## 1 -410.34
## 2 -406.23 8.2238 11     0.6931
```

As we might have suspected, none of the interaction terms are needed, so to avoid overfitting, we do not include the interaction terms in our final model.

Influential Observations

In brief, an influential observation is one that is an outlier (unusual time to failure given covariates) and has leverage (an unusual observation in the x-direction). This has the effect of strongly influencing β. To check influence, I'm using dfbeta values which measures the change in β when a purported influential point is removed.

```
#plot dfbeta plot
ggcoxdiagnostics(fit4, type = "dfbeta",
                 linear.predictions = FALSE, ggtheme = theme_bw(),
                 title = "Figure 6. Plot of Dfbeta values for each explanatory variable")
```


Figure 6. Plot of Dfbeta values for each explanatory variable
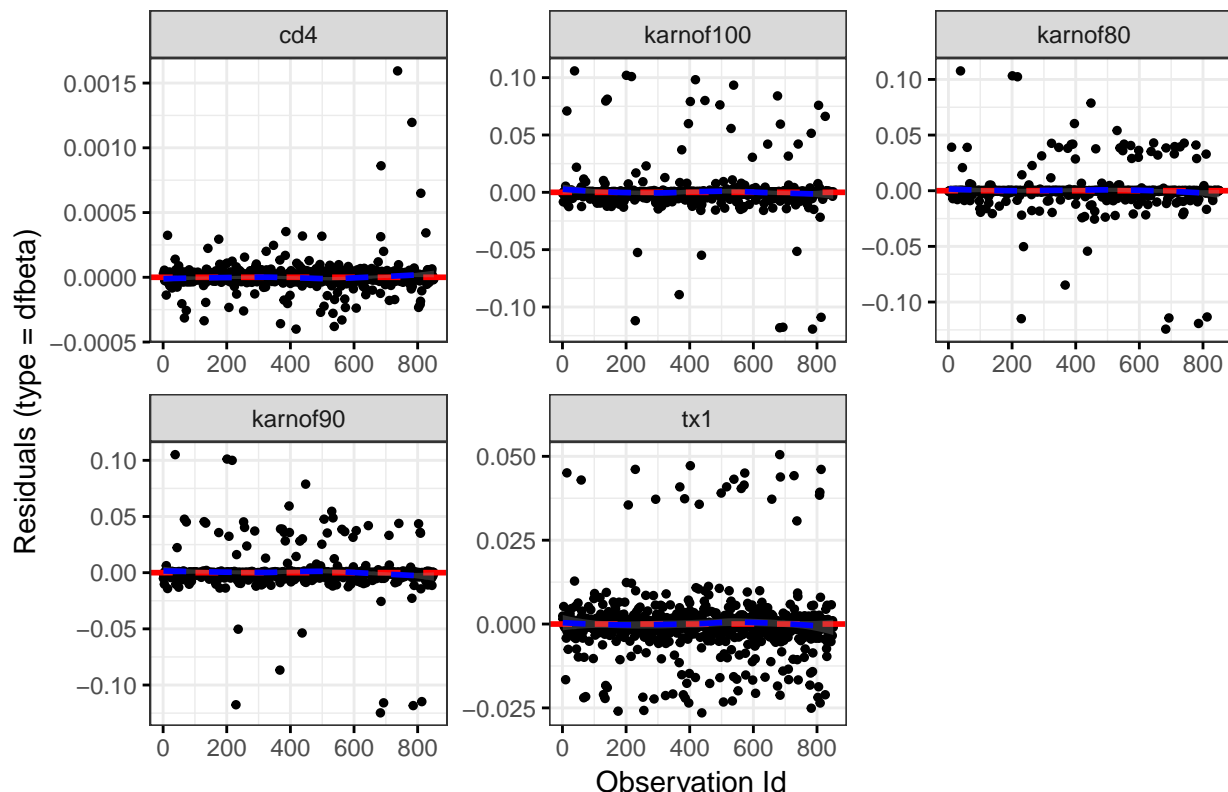
The above index plots show that comparing the magnitudes of the largest dfbeta values to the regression coefficients suggests that none of the observations is super influential, even though some of the dfbeta values for `cd4` and `tx` are large compared with the others. Generally, we should be careful removing influential observations and throwing away data unless there's a *clear* reason we should (e.g. poor data entry), which we didn't find when checking the influential points for `cd4`. If we look at the deviance residuals for outliers, we

might initially be concerned because the distribution does not seem symmetric, with many patients having negative residuals that mean they "lived too long".

```
ggcoxdiagnostics(fit4, type = "deviance",
                 linear.predictions = FALSE, ggtheme = theme_bw(),
                 title = "Figure 7. Plot of Deviance for Final Model")
```

Figure 7. Plot of Deviance for Final Model
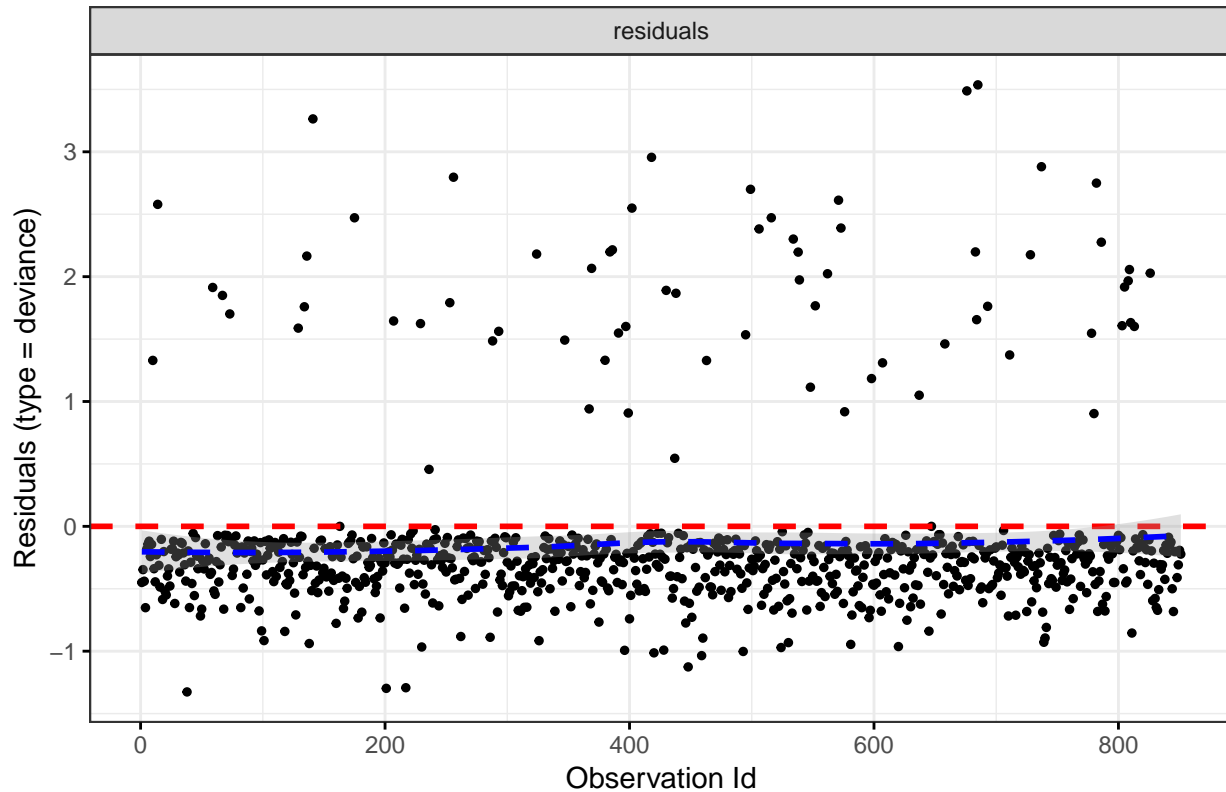
But then, we remember that our data was imbalanced between those who lived and died to begin with, because the majority of patients lived until the last time point. Thus, we are not too concerned with influential observations adversely affecting our model.

Checking the Log-Linearity Assumption

In class, we investigated the log-linearity assumption which basically verifies whether a continuous predictor variable is linearly correlated with our log hazard ratio. To do this, we first assume our continuous predictor is categorical, and then we take ratios between the instantaneous relative risks. If the ratios are all roughly the same, then we can assume a linear relationship. In our final model, the only continuous predictor we have is `cd4`, so lets categorize it in the following way:

Level	Desc
0-70	low (0)
71-140	low-medium (1)
141-210	medium (2)
211-280	medium-high (3)
281-350	high (4)

12

```
#turn cd4 into categorical
data2 <- data %>%
  mutate(cd4_group = ifelse(cd4 <= 70, 0,
                 ifelse(cd4 > 71 & cd4 <= 140, 1,
                      ifelse(cd4 > 141 & cd4 <= 210, 2,
                          ifelse(cd4 > 211 & cd4 <= 280, 3,4))))) %>%
  mutate(cd4_group = as.factor(cd4_group))
#fit our model with the categorical cd4 instead of the continuous one
fit5 <- coxph(Surv(time, censor) ~ tx + karnof + cd4_group, data = data2)
fit5 %>% tidy()
```

```
##          term    estimate    std.error     statistic       p.value  conf.low
## 1          tx1  -0.6800727    0.2572007  -2.644132153 0.0081900673 -1.184177
## 2    karnof80  -0.5807547    0.4120730  -1.409348998 0.1587319970 -1.388403
## 3    karnof90  -1.2995531    0.4090965  -3.176641956 0.0014899085 -2.101368
## 4   karnof100  -1.7502933    0.4599903  -3.805065638 0.0001417666 -2.651858
## 5   cd4_group1  -1.4899999    0.4333253  -3.438525292 0.0005848919 -2.339302
## 6   cd4_group2  -2.8400353    1.0102344  -2.811263621 0.0049347335 -4.820058
## 7   cd4_group3 -17.4248428 3031.6337920  -0.005747674 0.9954140449      -Inf
## 8   cd4_group4  -0.1441065    0.7219043  -0.199619878 0.8417778799 -1.559013
##     conf.high
## 1  -0.1759686
## 2   0.2268936
## 3  -0.4977387
## 4  -0.8487289
## 5  -0.6406980
## 6  -0.8600122
## 7         Inf
## 8   1.2708000
```

Now we do some calculations. We want to see whether $ln(\frac{h_{cd4+100}(t)}{h_{cd4}(t)}) = 70\beta$ turns out to be relative constant. Between group0 and group1 of cd4, the log hazard ratio is $ln(\frac{e^{-1.49}}{e^0}) = -1.49$. Between group1 and group2 of cd4, the log hazard ratio is $\frac{e^{-2.84}}{e^{-1.49}} = -1.35$. We can already see that the coefficients for group3 and group4 are whack, which is probably due to a small number of observations within those categories. We conclude that from the similar log hazard ratios calculated above that it's probably safe to say that `cd4` can be modeled as a continuous linear predictor.

Validating our Model by Bootstrapping (Lathan)

Challenges

Personally, my biggest challenge when learning something new is deciding to what degree I'd like to understand the topic. There is a surface understanding of the definition, a more difficult understanding of the mathematics, and an even more difficult understanding of the conceptual applications. In the case of bootstrap, I think I will find challenging understanding the math behind how bootstrap works.

A brief overview of Bootstrap and its applications to survival analysis

Bootstrap relies on sampling with replacement of the sample data and in the case of modelling, it is used to evaluate the performance of the model on the original sample. The estimate of the likely performance of the final model on future data is estimated by the average of all the indices computed on the original sample. If

we had an original sample of n elements,X, we resample X m times to get new bootstrap samples $X_i, ... X_m$ each with size n, derive a model in the bootstrap sample, and apply it to the original sample.

Bootstrapping validates the *process* of obtaining our original Cox PH model. It also tends to provide good estimates of the future performance of our final model if the same modeling process was used in our bootstrap samples. One of the strengths of bootstrapping is thati can estimate the bias due to overfitting in our final model - let's call this quantity "optimism". You can subtract from the original sample estimate the "optimism" to get the bias-corrected estimate of predictive accuracy.

```
#add data to model fit so bootstrap can re-sample
final.fit <- cph(Surv(time, censor) ~ tx + karnof + cd4, data = data)
g <- update(final.fit, x = TRUE, y = TRUE)
set.seed(47)
#bootstrap validation
validate(g, B = 300)
```

```
##          index.orig training   test optimism index.corrected   n
## Dxy          0.5826   0.5848 0.5689   0.0159          0.5667 300
## R2           0.1295   0.1358 0.1228   0.0129          0.1166 300
## Slope        1.0000   1.0000 0.9452   0.0548          0.9452 300
## D            0.0822   0.0864 0.0777   0.0087          0.0735 300
## U           -0.0022  -0.0022 0.0015  -0.0037          0.0015 300
## Q            0.0844   0.0886 0.0762   0.0124          0.0720 300
## g            1.4159   1.4555 1.3558   0.0997          1.3162 300
```

Training here is defined as the accuracy when evaluated on the bootstrap sample and test is when the model is applied to the original sample. Our D_{xy} is 0.5632 which is the difference between the probability of concordance and the probability of discordance of pairs of predicted survival times and pairs of observed survival times. This is essentially a measure of our accuracy of our model on new data.

Validating the SE and the Coefficients via Bootstrap

One idea of validating the standard error and the beta coefficients that we obtained in our original coxph model is to compute our bootstrap standard errors and beta coefficients and view their distributions. In the case of standard errors, if the bootstrap standard errors are close to the original standard errors, then we feel good about the variability of our original estimate. Similarly, if the boostrapped sampling distribution of our betas seems "normal" around our original beta coefficient, then we feel good about our estimate because we are comfortable that our estimated coefficients were close to the bootstrapped average over many new samples.

A little note about the boostrap methodology we used: it was a fairly straightforward sample with replacement without *fixing* the proportion of censored data. In other words, in each bootstrap sample, it could vary from 700 censored events to say, 710 censored events. Future exploratory directions could include trying other bootstrap sampling methods, such as separately bootstrapping censored and event data.

We first compare the standard errors between our bootstrap coefficients and our likelihood coefficients from our coxph model.

```
comparison
```

```
## # A tibble: 5 x 3
##    likese bootavgse coef
##     <dbl>     <dbl> <chr>
## 1 0.258      0.261  tx1
## 2 0.412      0.437  karnof80
## 3 0.413      0.436  karnof90
## 4 0.465      0.493  karnof100
```

```
## 5 0.00308    0.00313 cd4
```

We see that the standard errors are pretty similar! We also provide a boxplot to visualize how the distribution of bootstrapped standard errors compare to our original standard error estimates.
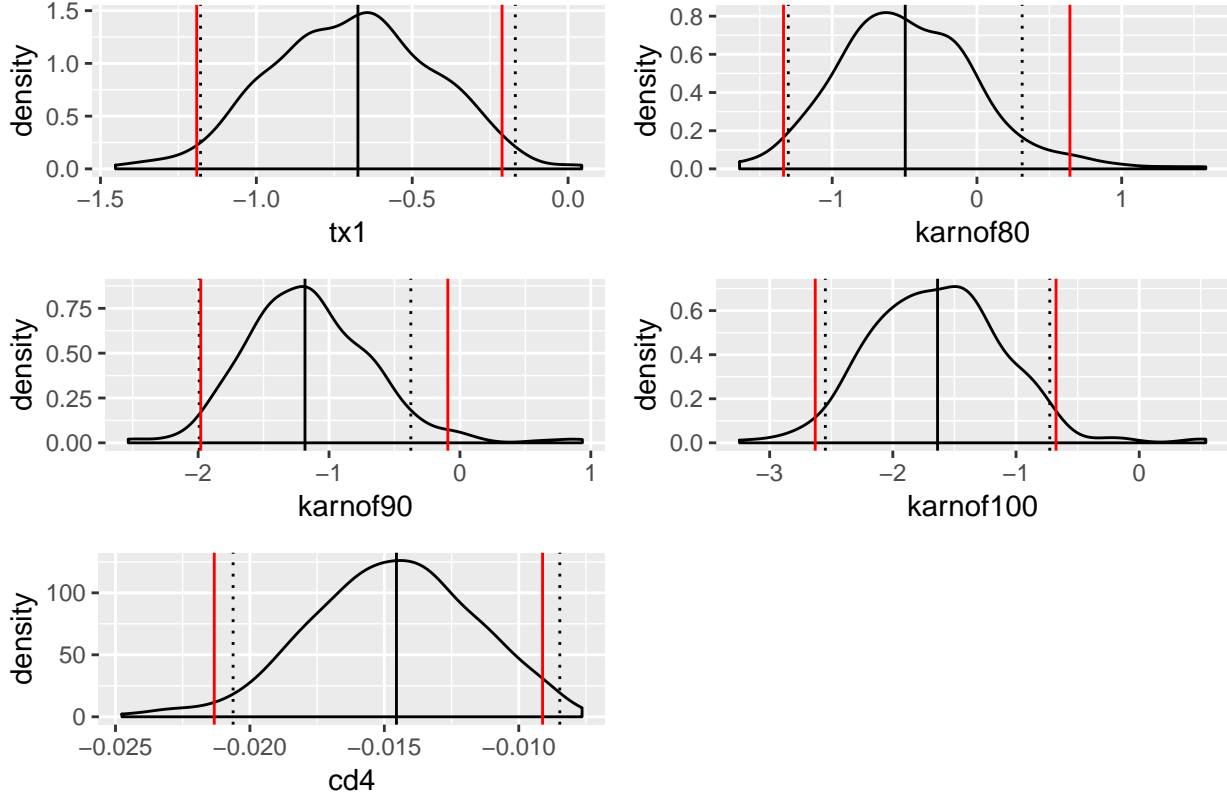
Figure 8. Distribution of Bootstrapped SE

Original SE (blue). Standard deviation of bootstrapped SE (red).
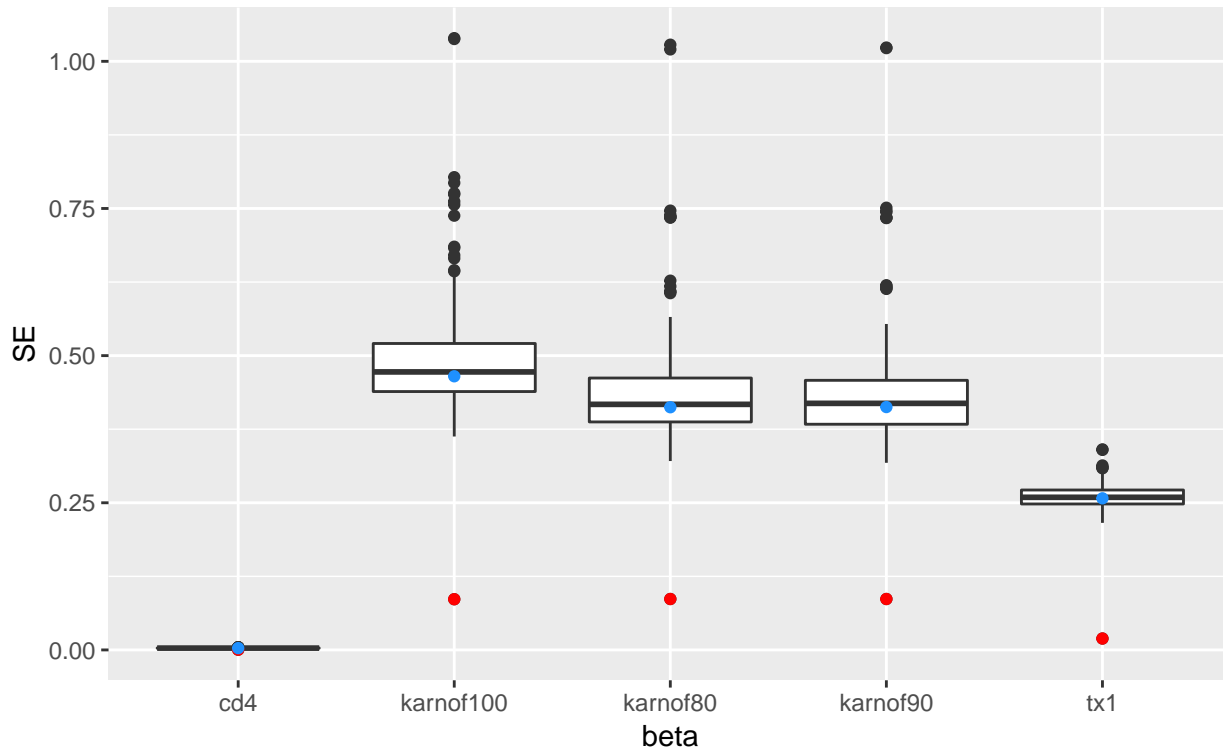
Further, we explore how the bootstrap betas vary compared to our original betas.

Figure 9. Bootstrap Sampling Distribution Beta Coefficients

As we can see, the sampling distribution of the bootstrapped betas fall more or less "normally" around the original estimate (shown by the solid black line). The 95% confidence intervals of the original estimates are displayed by dotted lines whereas the 95% confidence intervals of the bootstrap distributions are displayed by solid red lines. The confidence intervals are fairly similar, with `karnof80` and `karnof90` being slightly right skewed. Thus, I feel comfortable about the original estimates of the coefficients in that they weren't estimates at the tail end of a theoretical sampling distribution approximating a "true" distribution.

Interpreting our final model

The final model obtained after selecting features via LASSO was

$$ln(\frac{h_i(t)}{h_0(t)}) = \beta_1 tx1 + \gamma_1 karnof80 + \gamma_2 karnof90 + \gamma_3 karnof100 + \theta_1 cd4$$

. To better understand the effects of each predictor, we provide the following graphics.
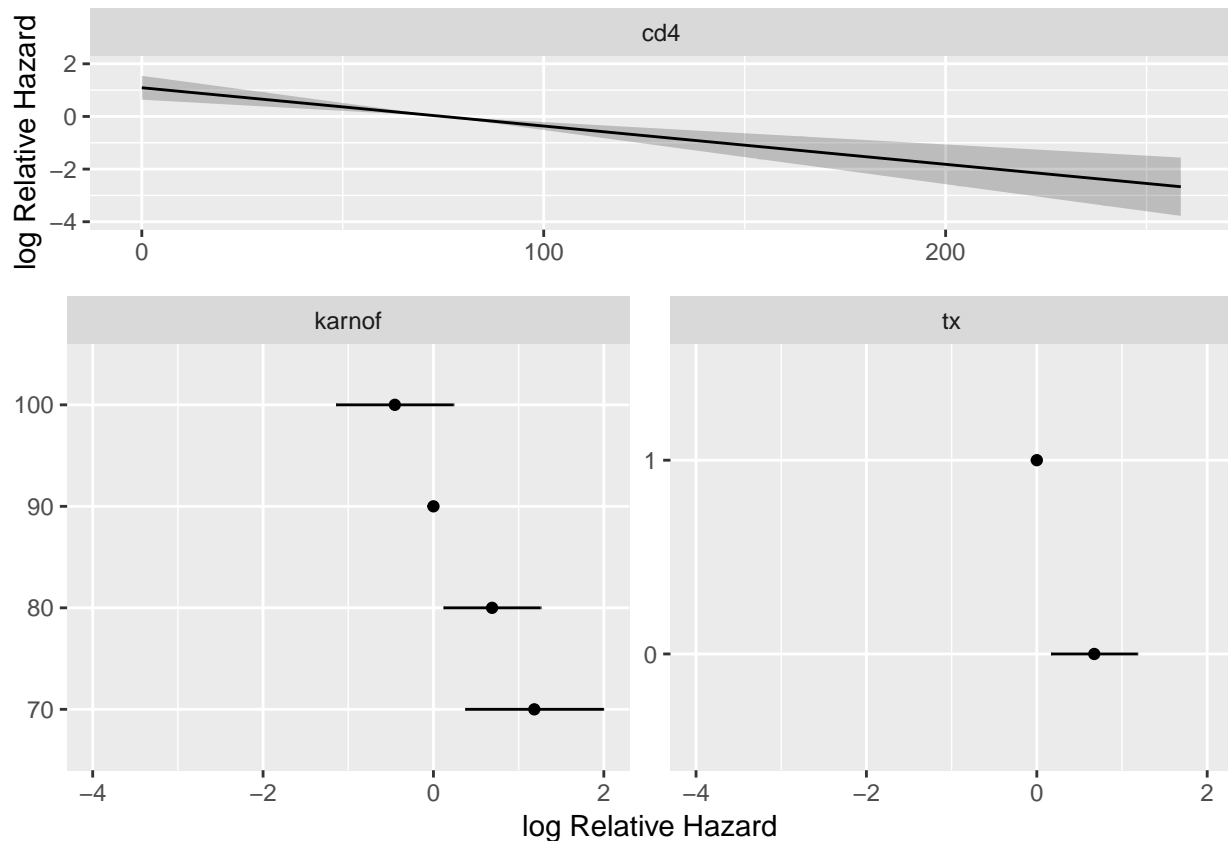
Figure 10. The Effects of Each Predictor on Log Survival Time

This plot shows the effect of each predictor on log survival time. Predicted values have been centered. 95% confidence intervals are also shown for the continuous variables. We observe that as cd4 count increases, the log of the relative hazard decreases. As Karnofsky score decreases, the log of the relative hazard goes up, which is to be expected. It seems that the treatment group including IDV is associated with a lower log relative hazard.

Figure 11. The Estimated Change in Median Survival Time for Each
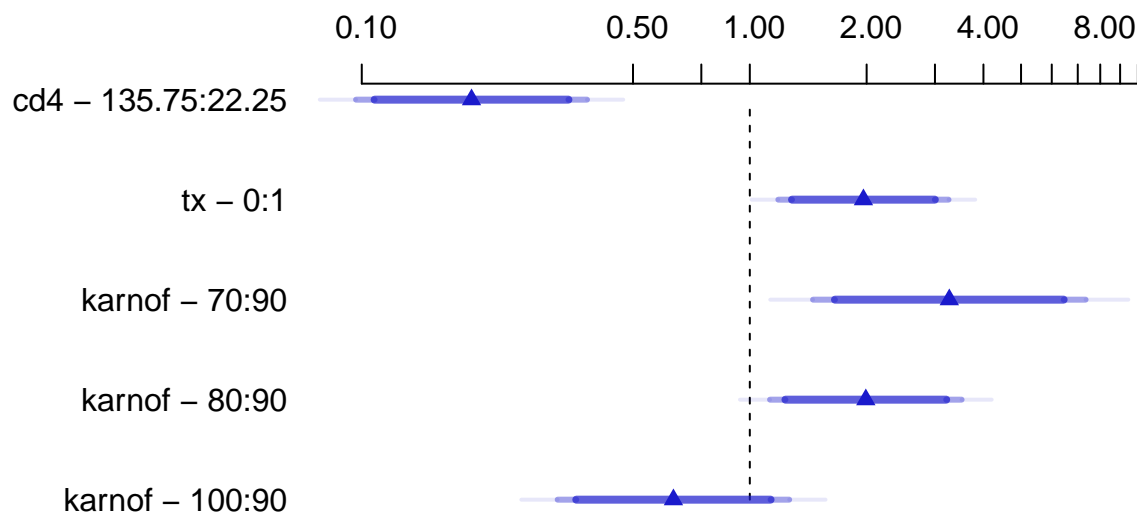
This above plot portrays the estimated change in median survival time for each predictor. Different shaded areas of bar indicate different confidence levels (.9, 0.95, 0.99). We see that as the cd4 count goes from 22.25

to 135.75, the median survival time decreases by more than half. Or when the Karnofsky score goes from 70 to 90, we observe a three-fold increase in median survival time.

Gradient Boosted Trees (Madison)

In addition to running a CoxPH model, we decided to try running a Cox Proportional Hazards model via a gradient boosted tree.

Background

Gradient boosting machines have gained traction in recent years, popular among Kagglers, researchers, and industry professionals alike. One of the publically availabe algorithms that has fueled this trend is XGBoost (eXtreme Gradient Boosting) developed by Tianqi Chen. XGBoost claims to be a scalable, high-performing, and one of the most computationally efficient implementations of gradient boosting machines out there. It can be used for a variety of regression, classification, and ranking problems.

Gradient boosting is a supervised ensemble method which agglomerates simple, "weak" learners into a more complex whole. In boosting (also called additive training), we start with a constant prediction and iteratively add new functions on top, fixing what we have learned and adding one new model at a time will holding onto functions learned in previous rounds. We fit each model to new residuals based on the previous prediction, then minimize the loss with the addition of the latest prediction. Thus, the residuals from each previous round are used to train the model. In doing this, we are actually updating our model each time using gradient descent - hence the name "gradient" boosting! Gradient boosting is possible with almost any simple classifier, but XGBoost in particular uses an ensemble of decision trees. The objective function within XGBoost also incorporates a regularization term.

By default, XGBoost in Python has mean squared error as its loss funciton within its objective function. However, the creators of XGBoost recently added the option to instead use the Cox regression loss function for right-censored survival time data, and that is what I will be using. Predictions are then returned on the hazard ratio scale. The package also includes the negative partial log-likelihood for Cox proportional hazards regression as an evaluation metric.

Cox Proportional Hazards in Generalized Boosted Models

In 2018, Tianqi Chen and other contributors added survival analysis cabability to XGBoost. They use gradient boosting with "the loss function from penalized Cox partial likelihood..., where regularization is explicitly imposed through penalization" ("Boosted nonparametric hazards with time-dependent covariates", Lee et al.). Since the paper associated with XGBoost was published before their survival analysis add-on, I relied on previous publications to give the best explanation of the Cox algorithm underying XGBoost. Their work is summarized below.

But first, to facilitate the understanding of notation downstream, recall the Cox proportional hazards model. For observations $(t_i, \delta_i, x_i), i = 1, \ldots, n$ where t_i is the observed time to event for individual i and δ_i is 1 if an event occurred at time t_i and 0 if the observation has been censored, and $x_i = (x_{i1}, \ldots, x_{ip})$ is a vector of covariates obtained at time 0. Then the hazard function is $h(t|x_i) = h_0(t) \exp(F(x_i; \beta))$ where $h_0(t)$ is the unspecified baseline hazard and $F(x; \beta)$ is a function of the covariates dependent on a parameter vector β. We like to use a linear predictor of the form $F(x; \beta)$ where each element of the β vector describes the influence of a single covariate. We can obtain an estimate for β by maximizing the partial log-likelihood:

$$l(\beta) = \sum_{i=1}^{n} \delta_i (F(x_i; \beta) - \log(\sum_{j:t_j \geq t_i} \exp(F(x; \beta))))$$

The Cox additions to XGBoost were inspired by the existing Generalized Boosted Models R package `gbm` (Greenwell et al.) which has included the Cox proportional hazard model since 2007. In "Generalized Boosted Models: A guide to the gbm package," Greg Ridgeway details how the developers implementated the Cox proportional hazard model. According to Ridgeway, "The Cox proportional hazard model... is an incredibly useful model and the boosting framework applies quite readily with only slight modification" (Ridgeway 1). Chen et al. detail the mathematics in the context of gradient-boosting trees, summarized below.

The goal of any GBM (gradient boosting machine) is to learn a functional mapping from the data $\{x_i, y_i\}_{i=1}^n$ to $y = F(x, \beta)$ where β is the set of parameters of the function F which minimize some cost function $\sum_{i=i}^n \Phi(y_i, F(x_i; \beta))$. An important assumption of boosting is that $F(x) = \sum_{m=0}^M \rho_m f(x; \tau_m)$ where f is a "weak" learner with a weight ρ and set parameter τ_m (described just below). In other words, this assumption means that $F(x)$ follows an "additive" expansion form, permitting us to perform "additive training" or boosting. Otherwise, there are no functional assumptions made on F, allowing it to be linear, tree-based, or otherwise. Thus, the set of parameters of the function F is exactly $\beta = \{\rho_m, \tau_m\}_{m=1}^M$. These are learned in a greedy iterative process, detailed below:

1. Set an initial estimator $f_0(x)$

2. For each iteration $m \in \{1, \ldots, M\}$, solve:

$$(\rho_m, \tau_m) = \arg\min_{\rho, \tau} \sum_{i=1}^n \Phi(y_i; F_{m-1}(x_i) + \rho f(x_i; \tau))$$

We next approximate this second step by

   (a) Fit $f(x; \tau_m)$ by

$$\tau_m = \arg\min_\tau \sum_{i=1}^n (g_{im} - f(x_i; \tau))^2$$

   where

$$g_{im} = -[\frac{\delta\Phi(y_i, F(x_i))}{\delta F(x_i)}]_{F(x)=F_{m-1}(x)}$$

   (b) Learn ρ by

$$\rho_m = \arg\min_\rho \sum_{i=1}^n \Phi(y_i, F_m(x_i) + \rho f(x_i; \tau_m))$$

3. Update $F_m(x) = F_{m-1}(x) + \nu \rho_m f(x; \tau_m)$ for some learning rate or shrinkage term, $\nu \in (0, 1]$ (this is called 'eta' in the XGBoost code). The shrinkage term is used to control overfitting and typically is valued between $0.01 - 0.2$. The complexity of $f(x)$ is informed by the tree depth and minimum number of samples in terminal nodes. We can also improve GBM performance by subsampling, or fitting each weak learner on a random subset of the training data, known as stochastic gradient boosting.

To boost a proportional hazards model, the cost function becomes the negative partial log-likelihood:

$$\phi(y, F) = -\sum_{i=1}^n \delta_i (F(x_i) - \log(\sum_{j:t_j \geq t_i} e^{F(x_j)}))$$

.

Inspired by Binder and Schumacher's `CoxBoost` algorithm detailed in "Allowing for mandatory covariates in boosting estimation of sparse high-dimensional survival models" (2008), the XGBoost developers include a penalization term λ in the cost function Φ. This penalty term λ helps avoid overfitting, as the higher it is, the more cautious the update.

Analysis

See the XGBoost jupyter notebook for the analysis (I was really struggling in getting Python to work in RStudio, so putting mycode in this markdown feels like more trouble than it is worth right now!).

Discussion and Concluding Remarks

Overall, our model points to treatment, Karnofsky score, and cd4 levels as important predictors for predicting time to death (or diagnosis). As seen in Figure 10, an increase in cd4 is associated with a statistically significant decrease in the log hazard ratio, which is sensible since cd4 cell count is a well-regarded indicator for the presence of HIV – the lower the cd4 count, the higher the activity of the HIV virus killing off these cells. As seen in Figure 11, when the Karnofsky score goes from 70 to 90, we observe a signficant three-fold increase in median survival time. This result may even serve to help validate the Karnofsky performance metric intrinsically. In addition, Figure 5, our model fit, Figures 10 and 11 all show that there is a significant difference between the control and treatment groups regarding survival, with the treatment group having a higher survival.

Thus, given that this was an experimental study, we conclude that the treatment containing IDV significantly increases survival after adjusting for cd4 levels and Karnofsky score. We are willing to generalize this claim to patients who have no more than 200 CD4 cells/cubic millimeters, at least 3 months of prior ZDV therapy, and who reside in the United States since those were the inclusion criteria for this study.

References

- https://homes.cs.washington.edu/ tqchen/pdf/BoostedTree.pdf
- https://xgboost.readthedocs.io/en/latest/tutorials/model.html
- https://slundberg.github.io/shap/notebooks/NHANES
- https://statisticalhorizons.com/multicollinearity (Great article on multicollinearity)
- Harrell, F. (2015) Regression Modeling Strategies. (great textbook on all things survival analysis)
- https://www.datacamp.com/community/tutorials/bootstrap-r (Overview of bootstrapping)
- https://stats.stackexchange.com/questions/22017/sample-size-and-cross-validation-methods-for-cox-regression-predictive-models
- https://stats.stackexchange.com/questions/18084/collinearity-between-categorical-variables (Explanation of VIF for Categorical Variables)
- https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/
- https://stackoverflow.com/questions/53562813/xgboost-cox-survival-time-entry
- https://www.analyticsvidhya.com/blog/2016/02/complete-guide-parameter-tuning-gradient-boosting-gbm-python/
- https://arxiv.org/pdf/1802.03888.pdf
- http://www.saedsayad.com/docs/gbm2.pdf
- https://arxiv.org/pdf/1701.07926.pdf
- https://www.hindawi.com/journals/cmmm/2013/873595/