# Cox PH Model and Something New

*Madison Hobbs & Lathan Liou*

*4/19/2019*

## Introduction

Our project seeks to understand 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 remaining explanatory variables through some exploratory data analysis. Next, we try a series of Cox Proportional Hazards models. Finally, we investigate extentions including Bootstrapping techniques and alternative model algorithms such as XGBoost.

## Exploratory Data Analysis

### 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), as shown below:

```
aids %>% group_by(txgrp) %>% summarise(n())
```

```
## # A tibble: 2 x 2
##   txgrp `n()`
##   <int> <int>
## 1     1   422
## 2     2   429
```

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 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.

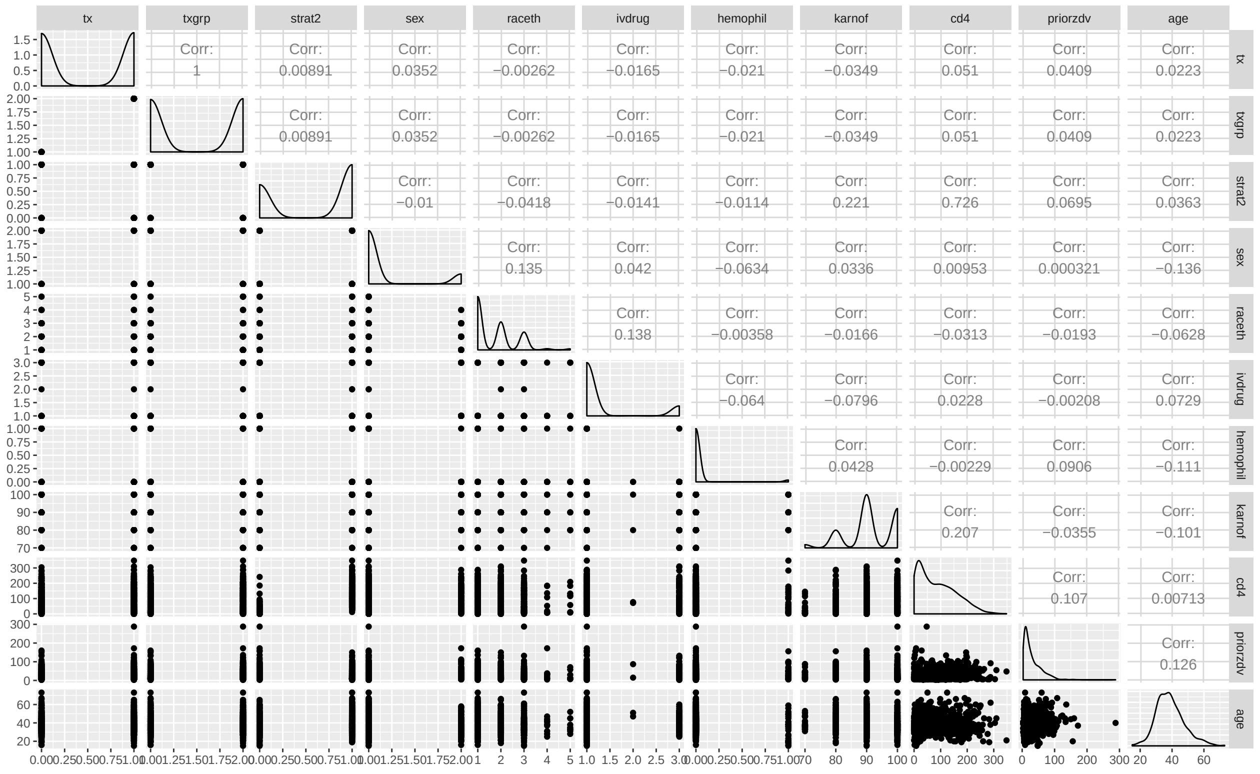
```
# Is it true that for every entry in `aids`
all(
  (aids %>%
     filter(((txgrp == 1 && tx == 0) || (txgrp == 2 && tx == 1))))
  == aids) == TRUE
```

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

### Correlation

We present a pairs plot of our explanatory variables, excluding id, our time-to-event variables, and our censoring variables to 1) visualize the distribution of the variables and 2) identify potential pair-wise correlation. `cd4` and `strat2` have a correlation coefficient of 0.74, which indicates moderate to strong correlation. This makes sense since `strat2` is the indicator variable for the continuous variable, `cd4`. Additionally, as noted just above, `tx` and `txgrp` provide the exact same information and therefore are 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).

```r
aids %>% select(tx:age) %>% ggpairs()
```



## 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.

```r
non_censored <- aids %>% filter(censor == 1) %>%
  mutate(tx=ifelse(tx == 0, "Control", "IDV"))
dim(non_censored) # complete AIDS or death
```

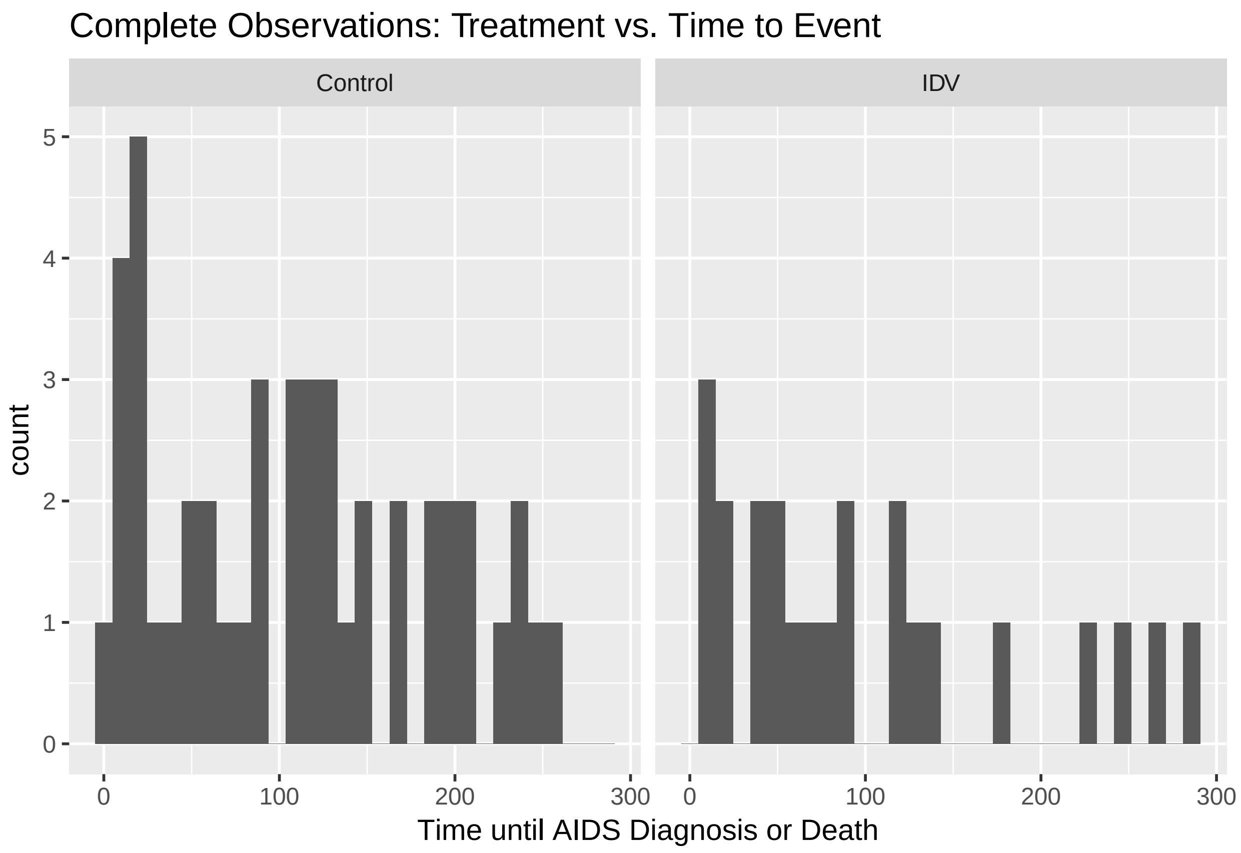
```
## [1] 69 16
```

```r
dim(aids)         # everyone
```

```
## [1] 851  16
```

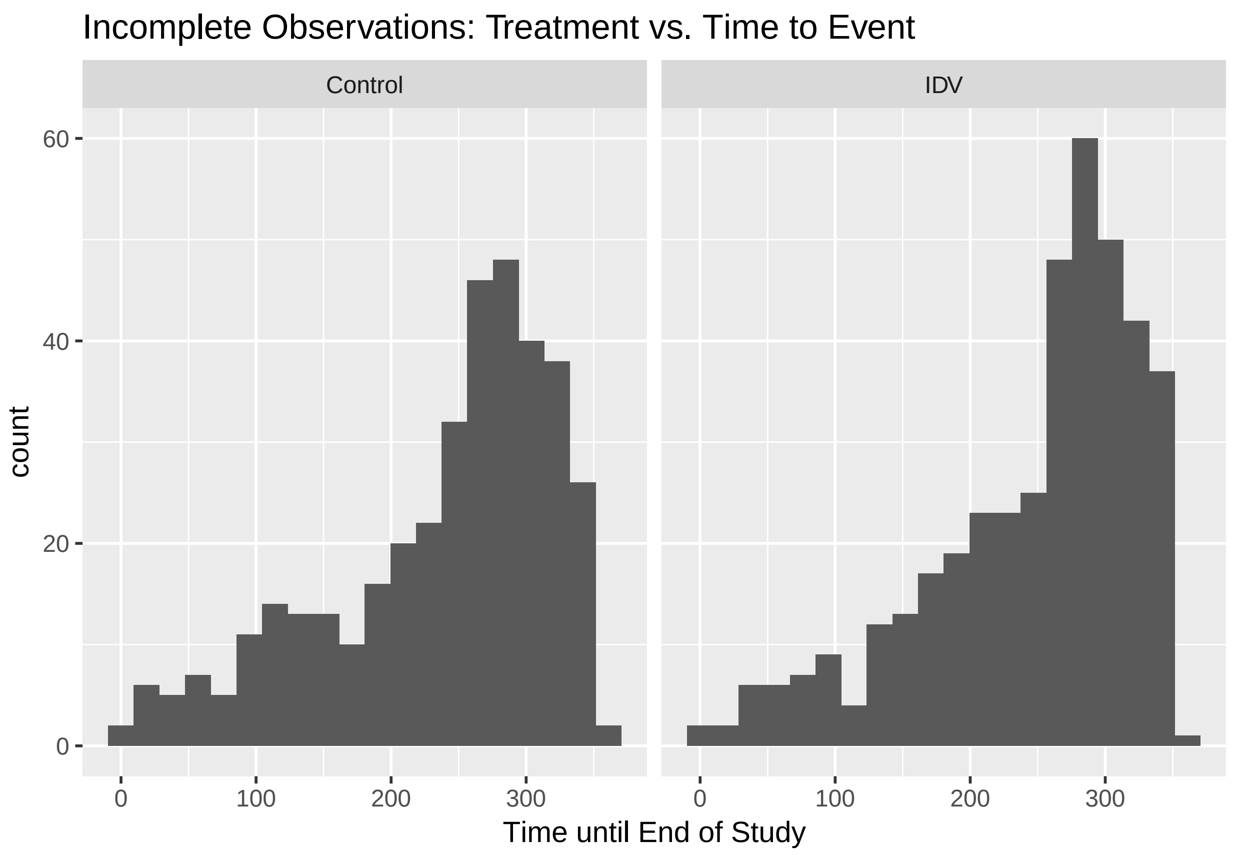
Among those with complete times, we notice from the side-by-side histograms below that both the control and IDV groups are skewed right. This makes sense - for complete observations, it's probably less common for people to last a long time without being diagnosed or dying. The distributions between the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

```r
ggplot(non_censored, aes(x = time)) +
  geom_histogram(bins = 30) +
  facet_grid(.~tx) + ggtitle("Complete Observations: Treatment vs. Time to Event") +
  xlab("Time until AIDS Diagnosis or Death")
```



When looking at the censored (incomplete) times for diagnosis/death, both control and IDV groups are in the opposite direction (left).

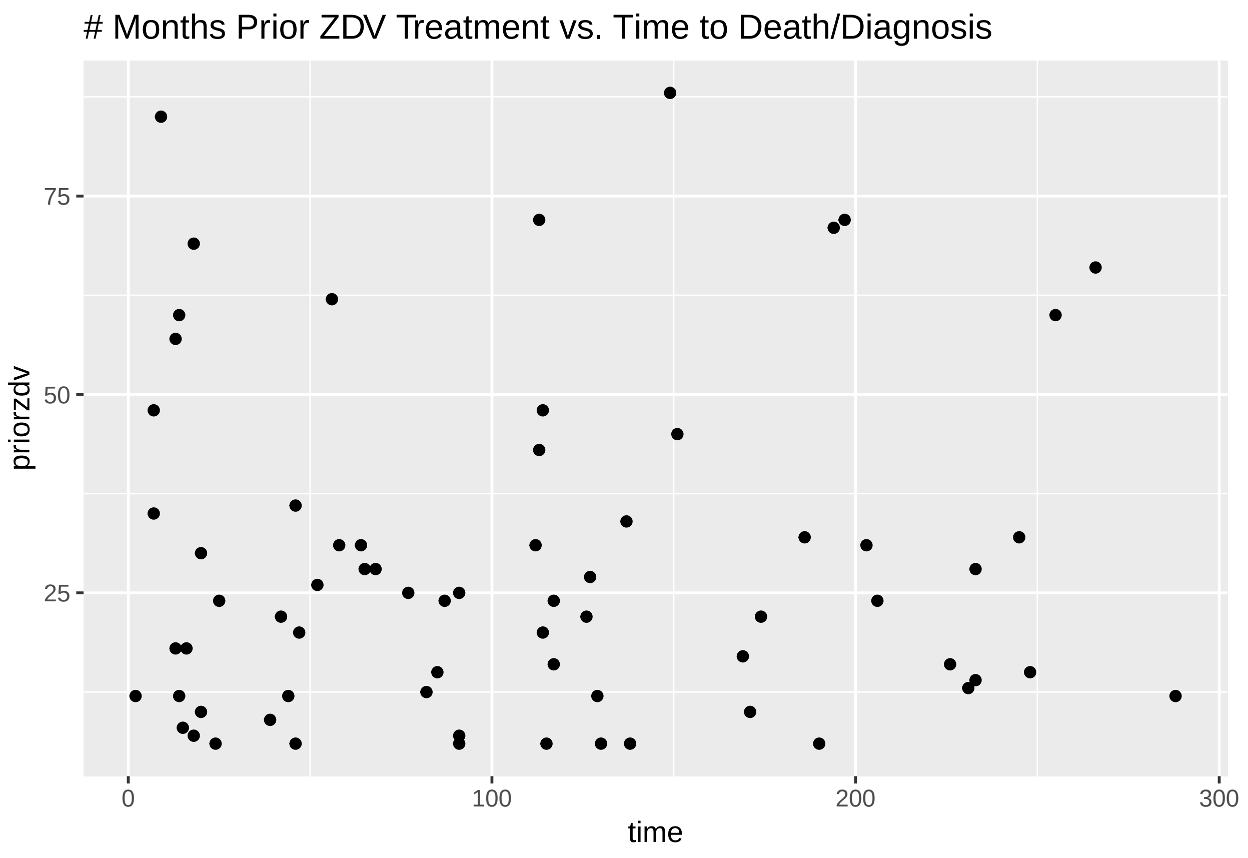
```r
ggplot(aids %>%
         filter(censor == 0) %>%
         mutate(tx=ifelse(tx == 0, "Control", "IDV")), aes(x = time_d)) +
  geom_histogram(bins = 20) +
  facet_grid(.~tx) +
  ggtitle("Incomplete Observations: Treatment vs. Time to Event") +
  xlab("Time until End of Study")
```



## Prior ZDV on Complete Observations

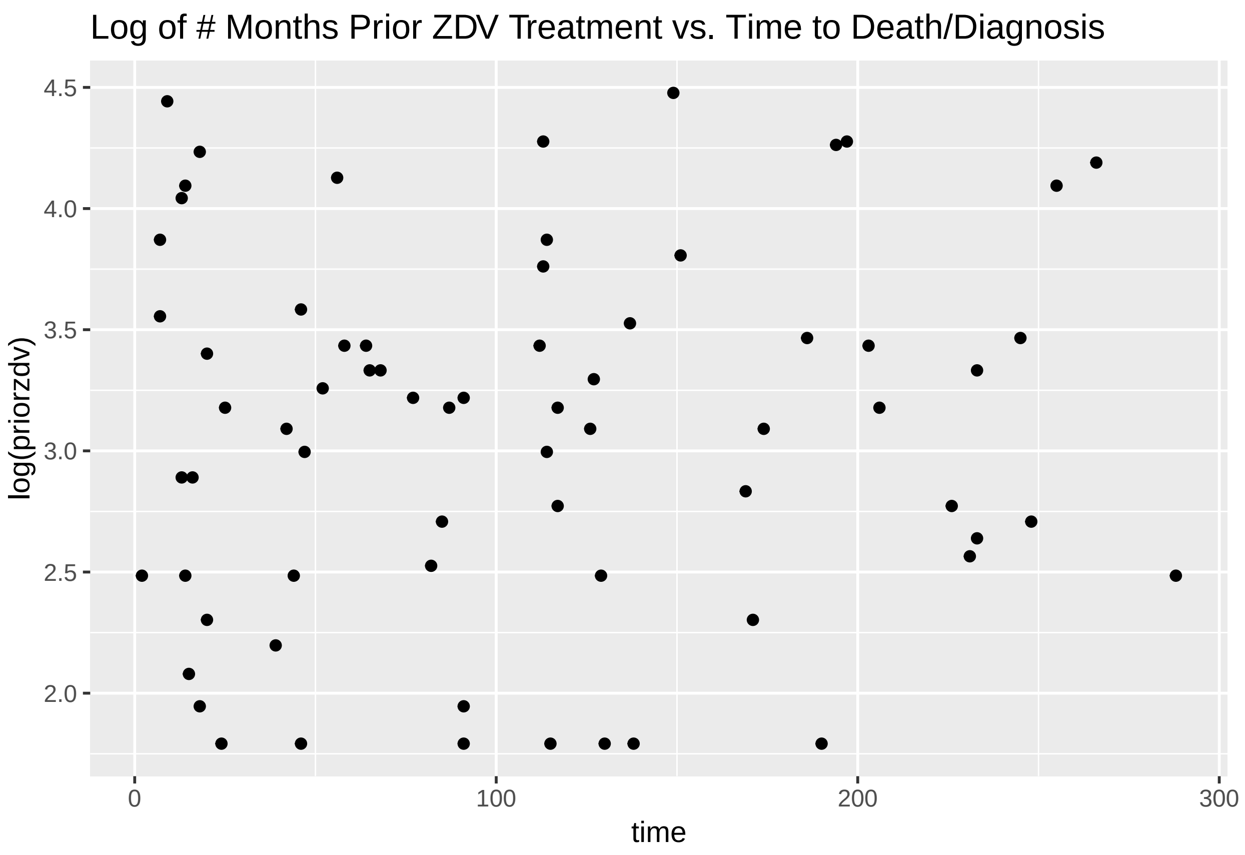
We were also curious about the relationship between time to diagnosis/death and number of months of prior ZDV use for non-censored participants. Interestingly, there appeared to be no relationship whatsoever, as evidenced by the following scatterplot:

```r
ggplot(non_censored, aes(x = time, y= priorzdv)) +
  geom_point() +
  ggtitle("# Months Prior ZDV Treatment vs. Time to Death/Diagnosis")
```



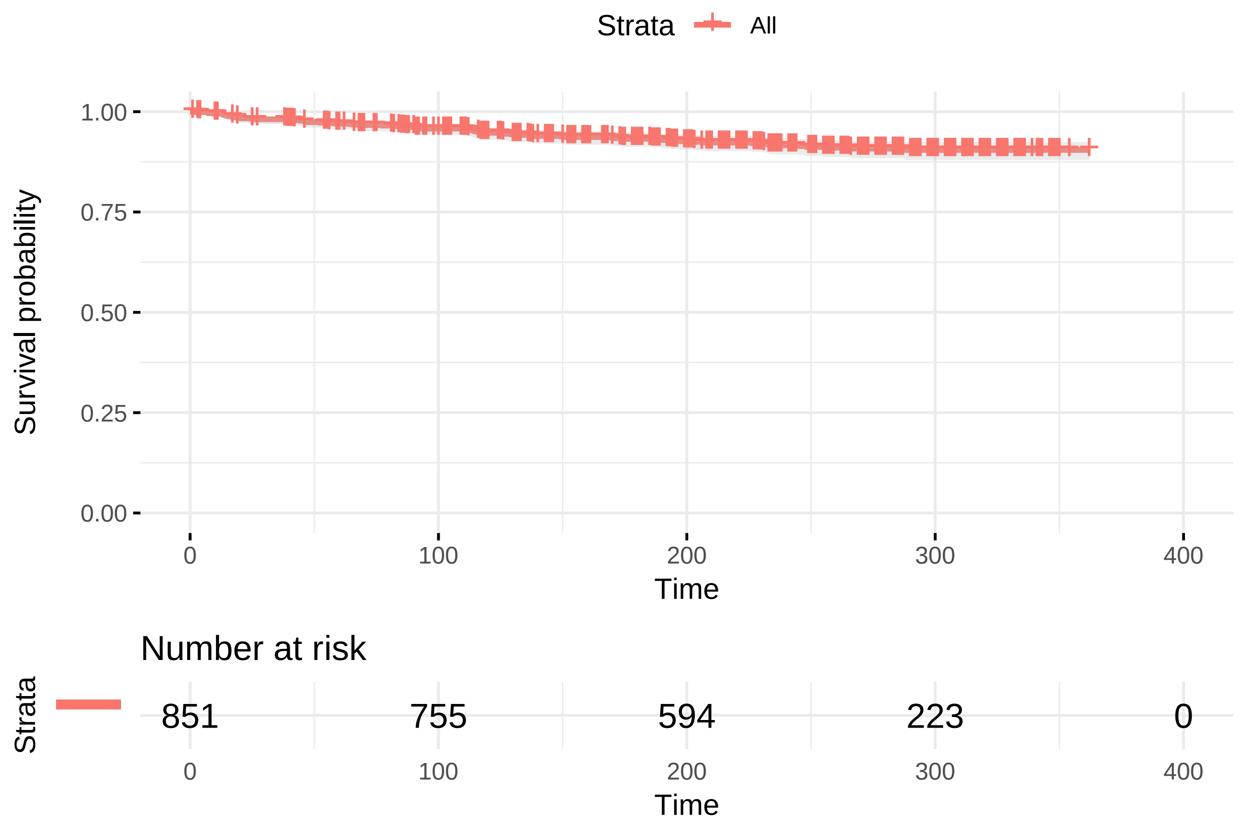
This finding is made even clearer when we log the number of months of prior zdv:

```r
ggplot(non_censored, aes(x = time, y= log(priorzdv))) +
  geom_point() +
  ggtitle("Log of # Months Prior ZDV Treatment vs. Time to Death/Diagnosis")
```



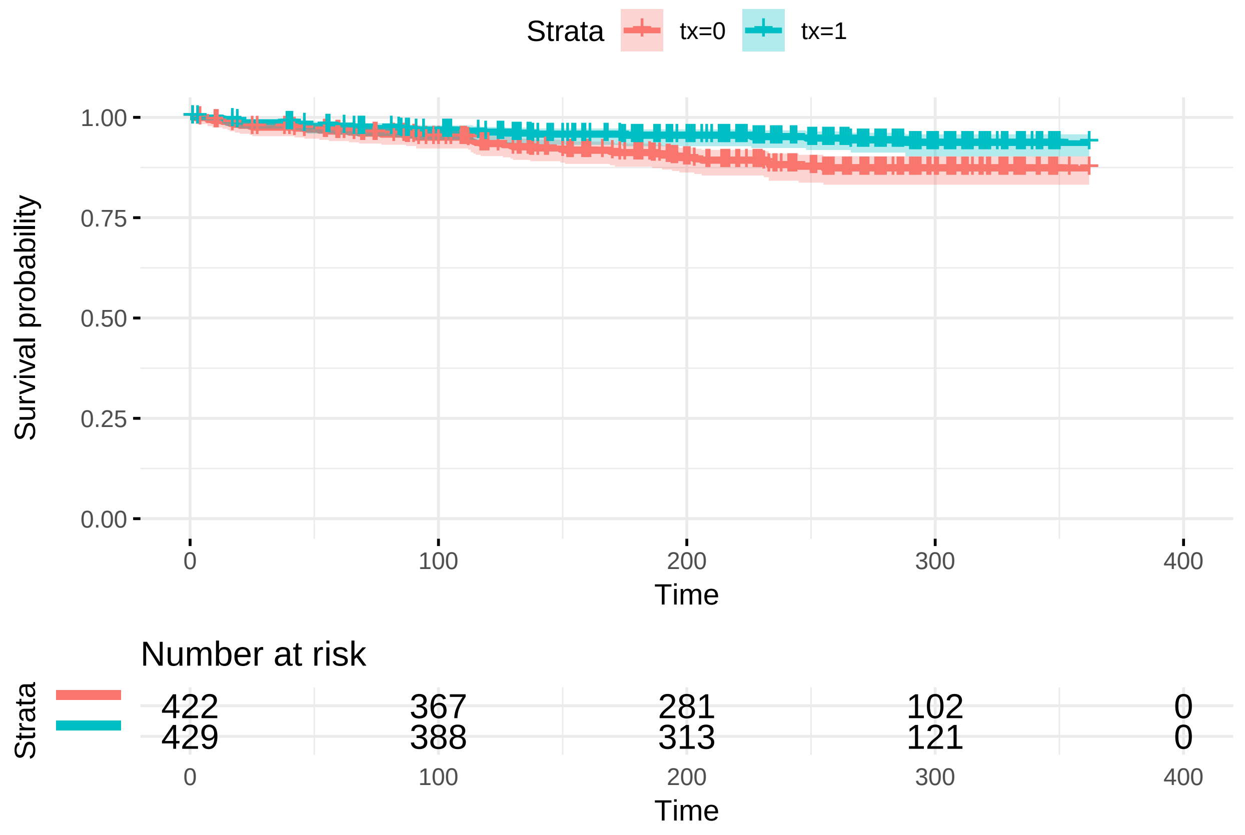
## Plotting Survival Curves

```
surv <- survfit(Surv(time, censor) ~ 1,
               data = data,
               type = "kaplan-meier",
               conf.typ ="log-log",
               se.fit = TRUE)
#plot KM curve
ggsurvplot(surv, data = data,
           risk.table = TRUE,
           conf.int = TRUE,
           ggtheme = theme_minimal(),
           risk.table.y.text.col = T,
           risk.table.y.text = F)
```



We first 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.

```
surv2 <- survfit(Surv(time, censor) ~ tx,
               data = data,
               type = "kaplan-meier",
               conf.typ ="log-log",
               se.fit = TRUE)

#plot KM curve
ggsurvplot(surv2, data = data,
           risk.table = TRUE,
           conf.int = TRUE,
           ggtheme = theme_minimal(),
           risk.table.y.text.col = T,
           risk.table.y.text = F)
```



```
#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 $\chi_1^2 = 9.2$ 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.

## 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. Since no p-values are significant at the $\alpha = 0.05$ level, we next build the full model.

```
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
```

```
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
```

We also checked whether any of the covariates were multicollinear with each other, and we see that none of them are multicollinear.

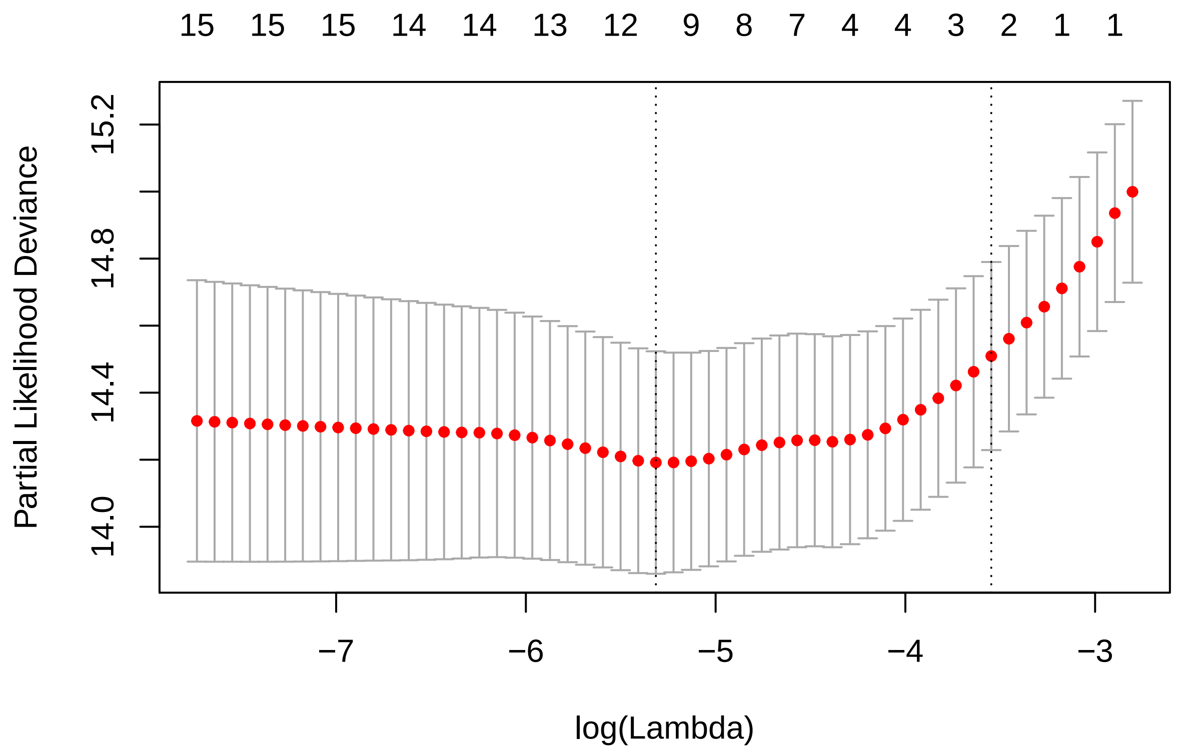
```r
#check multicollinearity
vif(fit)
```

```
##       tx1      sex2   raceth2   raceth3   raceth4   raceth5   ivdrug2
##  1.035589  1.083887  1.155469  1.136267  1.062312  1.111773  1.000000
##   ivdrug3 hemophil1  karnof80  karnof90 karnof100       cd4  priorzdv
##  1.077484  1.104453  2.820877  2.975839  2.240771  1.093961  1.063729
##       age
##  1.080429
```

We observe that none of the variables have a particularly high variance inflation factor (VIF). While `karnof80` and `karnof90` have relatively higher VIFs, we're not too concerned because they are dummy variables which necessarily have high VIFs due to the smaller proportion of cases in our reference category, `karnof70`.

The likelihood ratio $\chi^2$ statistic is 91.05 with 21 d.f. After considering whether variables can be mutating into new variables based on our conventional knowledge, and thinking of none in the moment, 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.

```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)

#plot cross-validated lambdas
plot(cv.fit)
```



```r
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 $\beta$ |
|---:|---:|
| (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, let's see what happens.

```r
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
```

Based on our likelihood ratio test results of a $\chi^2_1 = 0.2086$ and a p-value of 0.6479, priorzdv is not needed in the model.

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. Let's fit a model with only those 4 variables and conduct a likelihood ratio test between this model and the additive model without `priorzdv`.

```r
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 $\chi^2_8 = 6.72$ 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, let's try removing it from the model and seeing whether it's important or not.

```r
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 $\chi^2_1 = 2.23$ 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), let's interact treatment with our categorical covariates along with adjusting for cd4, priorzdv, and age.

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

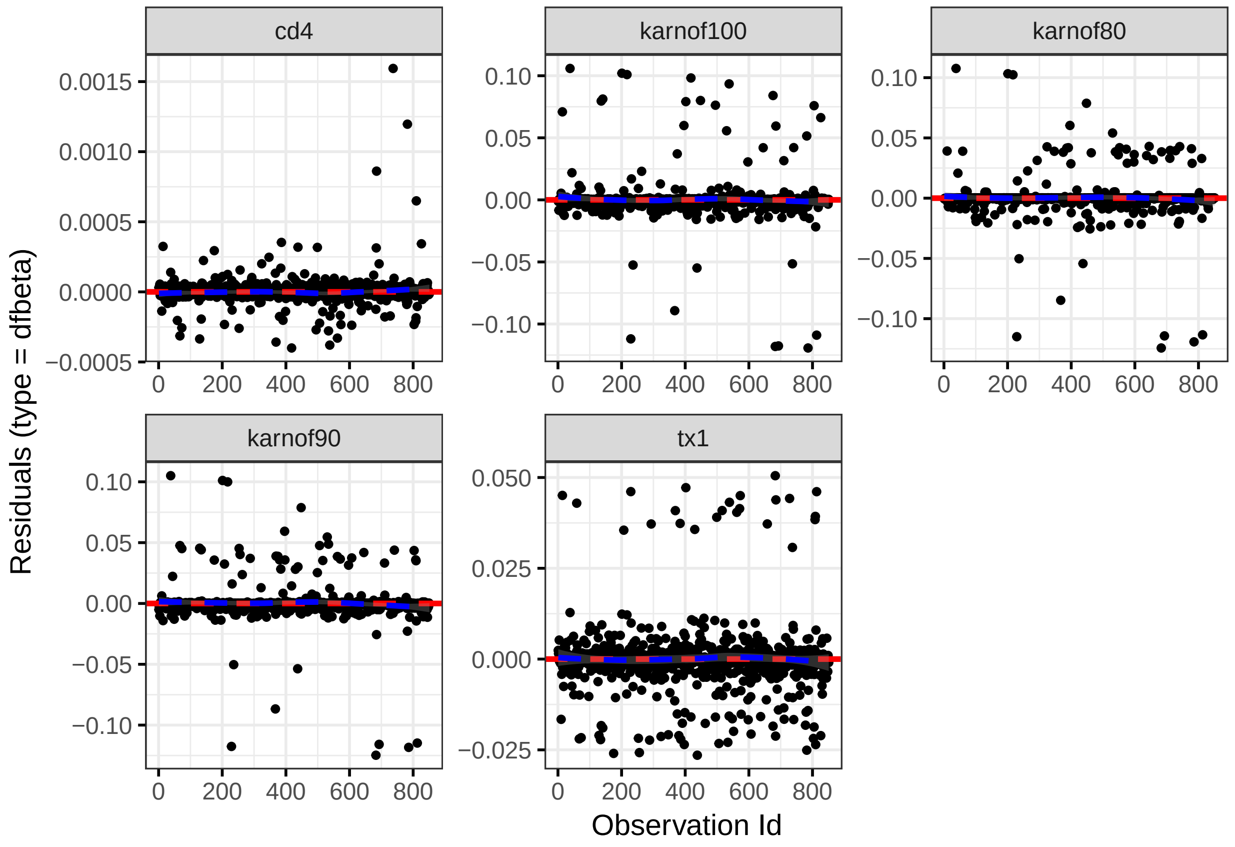
```
## Analysis of Deviance Table
##  Cox model: response is  Surv(time, censor)
##  Model 1: ~ tx + sex + raceth + ivdrug + hemophil + karnof + cd4 + priorzdv + age
##  Model 2: ~ tx * sex + tx * raceth + tx * ivdrug + tx * hemophil + tx * karnof + cd4 + priorzdv + ag
##    loglik  Chisq Df P(>|Chi|)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

As we might suspect, none of the interaction terms are needed, so to avoid overfitting, we won't 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 $\beta$. To check influence, I'm using dfbeta values which measures the change in $\beta$ when a purported influential point is removed.

```
#plot dfbeta plot
ggcoxdiagnostics(fit4, type = "dfbeta",
                 linear.predictions = FALSE, ggtheme = theme_bw())
```

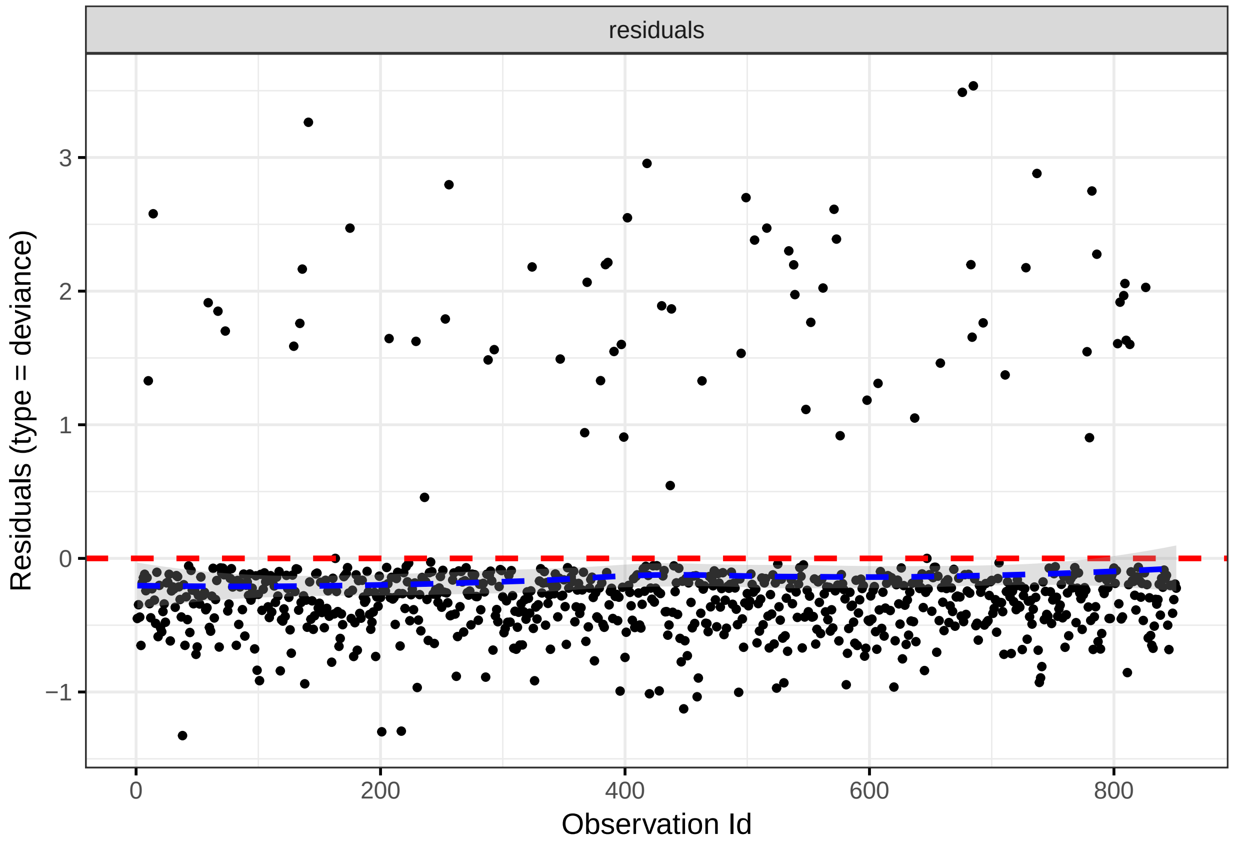


```
#identify influential points according to dfbeta
dfbeta <- residuals(fit4, type="dfbeta")

#check cd4 influential points
data[dfbeta[,5] > 0.0005,]
```

```
## # A tibble: 4 x 16
##       id  time censor time_d censor_d tx    txgrp strat2 sex   raceth
##    <int> <dbl>  <int>  <int>    <int> <fct> <fct> <fct>  <fct> <fct>
## 1    374    13      1    290        0 1     2     1      1     1
## 2   967   248       1     248         1 1      2      1       1      2
## 3   853   137       1     280         0 0      1      1       1      2
## 4   666   233       1     233         1 0      1      1       1      1
## # ... with 6 more variables: ivdrug <fct>, hemophil <fct>, karnof <fct>,
## #   cd4 <dbl>, priorzdv <dbl>, age <int>
```

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".

```r
ggcoxdiagnostics(fit4, type = "deviance",
                 linear.predictions = FALSE, ggtheme = theme_bw())
```



But then, we remember that our data was imbalanced between those who lived and died to begin with! The majority of patients lived until the last time point.

## 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) |

```
#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. I would conclude from the log hazard ratios calculated above (they're relatively similar) that it's probably safe to say that `cd4` can be modeled as a continuous predictor.

## Interpreting our final model

The final model obtained after selecting features via LASSO was

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

# 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.

```r
#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 can invoke some normal theory and be comfortable about the robustness of our estimated coefficients.

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.

Let's first compare the standard errors between our bootstrap coefficients and our likelihood coefficients from our coxph model.

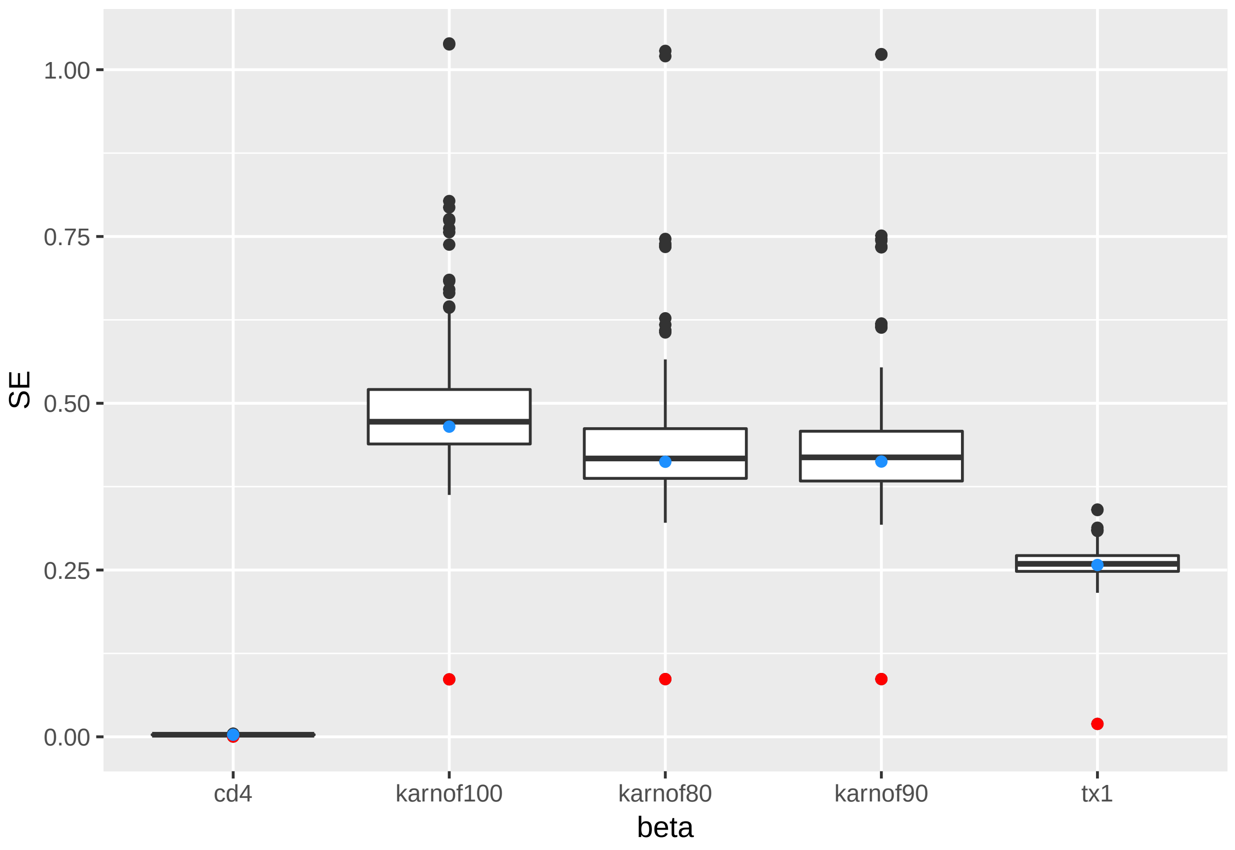
```
#initialize parameters
B <- 300
n <- nrow(data)

#initialize matrix
bootse <- matrix(NA, nrow = 1, ncol = 5)

#bootstrap model and obtain SE for each coefficient
set.seed(47)
for(i in 1:B) {
  j <- sample(1:n, n, TRUE)
  bootfit <- update(final.fit, data=data, subset=j)
  bootse <- rbind(bootse, as.vector(sqrt(diag(bootfit$var))))
}

#remove first row
bootse <- bootse[-1,]

#compute avg standard errors
bootavgse <- colMeans(bootse, 1)

#compare bootstrap standard errors and original standard errors
likese <- fit4 %>% tidy() %>% pull(3)
comparison <- tibble(likese, bootavgse, coef = c("tx1", "karnof80", "karnof90", "karnof100", "cd4"))
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! Here is a boxplot to visualize how the distribution of bootstrapped standard errors compare to our original standard error estimates.

```
#add sd of bootstrap
bootsd <- apply(bootse, 2, sd)
bootse2 <- rbind(bootse, bootsd)

#add original coxph SE
bootse3 <- rbind(bootse2, likese)

#turn to tibble
bootse_tb <- as_tibble(bootse3)
```

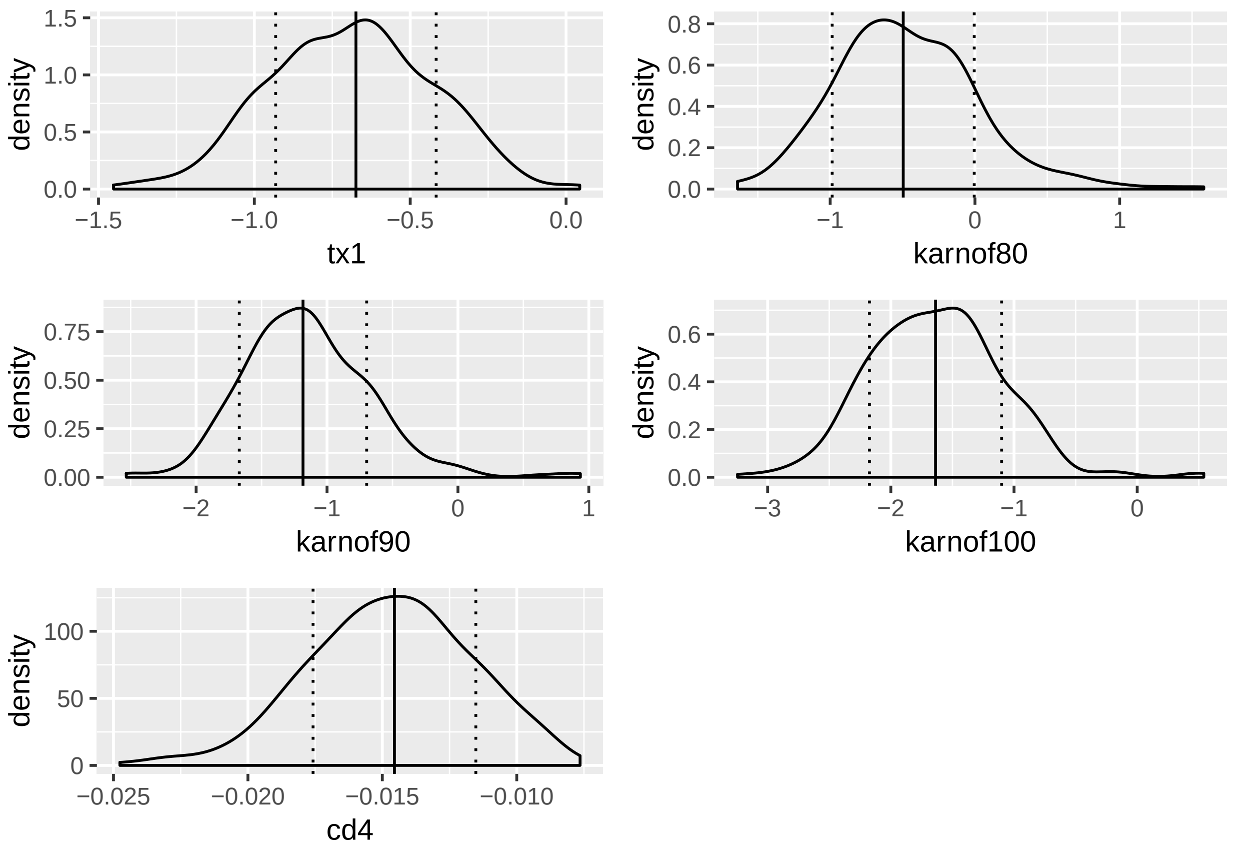
```
#tidy up bootstrap SE for plotting
colnames(bootse_tb) <- c("tx1", "karnof80", "karnof90", "karnof100", "cd4")
bootse_tb_tidy <- bootse_tb %>%
  gather(beta, value = tx1:cd4)

#mark standard deviations of bootstrap
bootse_tb_tidy <- bootse_tb_tidy %>%
  mutate(highlight = ifelse(row.names(bootse_tb_tidy) == 301 |
                              row.names(bootse_tb_tidy) == 603 |
                              row.names(bootse_tb_tidy) == 905 |
                              row.names(bootse_tb_tidy) == 1207 |
                              row.names(bootse_tb_tidy) == 1509, TRUE, FALSE))
#mark original coxph SE
bootse_tb_tidy <- bootse_tb_tidy %>%
  mutate(highlight2 = ifelse(row.names(bootse_tb_tidy) == 302 |
                              row.names(bootse_tb_tidy) == 604 |
                              row.names(bootse_tb_tidy) == 906 |
                              row.names(bootse_tb_tidy) == 1208 |
                              row.names(bootse_tb_tidy) == 1510, TRUE, FALSE))

ggplot(bootse_tb_tidy, aes(x = beta, y = `tx1:cd4`)) +
  geom_boxplot() +
  geom_point(data = subset(bootse_tb_tidy, highlight), #mark standard deviations in red
             aes(x = beta, y = `tx1:cd4`),
             color = "red") +
  geom_point(data = subset(bootse_tb_tidy, highlight2), #mark original coxph SE
             aes(x = beta, y = `tx1:cd4`),
             color = "dodgerblue") +
  labs(y = "SE")
```



Let's now explore how the bootstrap betas vary compared to our original betas.

```
#initialize matrix
bootbeta <- matrix(NA, nrow = 1, ncol = 5)

#bootstrap model and obtain betas
set.seed(47)
for(i in 1:B) {
  j <- sample(1:n, n, TRUE)
  bootfit <- update(final.fit, data=data, subset=j)
  bootbeta <- rbind(bootbeta, as.vector(bootfit$coefficients))
}

#remove first row
bootbeta <- bootbeta[-1,]

#convert into tibble
bootbeta_tb <- as_tibble(bootbeta)

#calculate sd for bootstrapped betas
bootbeta_sd <- apply(bootbeta_tb, 2, sd)
bootbeta_sd <- as.vector(bootbeta_sd)

#tidy up data
colnames(bootbeta_tb) <- c("tx1", "karnof80", "karnof90", "karnof100", "cd4")
bootbeta_tb_tidy <- bootbeta_tb %>%
  gather(beta, value = tx1:cd4) %>%
  mutate(value = `tx1:cd4`)
```

```r
#view densities of the each of the betas
p1 <- ggplot(bootbeta_tb, aes(tx1)) +
  geom_density() +
  geom_vline(xintercept = as.vector(final.fit$coefficients)[1]) +
  geom_vline(xintercept = c(as.vector(final.fit$coefficients)[1] - bootbeta_sd[1],
                            as.vector(final.fit$coefficients)[1] + bootbeta_sd[1]), linetype = "dotted"

p2 <- ggplot(bootbeta_tb, aes(karnof80)) +
  geom_density() +
  geom_vline(xintercept = as.vector(final.fit$coefficients)[2]) +
  geom_vline(xintercept = c(as.vector(final.fit$coefficients)[2] - bootbeta_sd[2],
                            as.vector(final.fit$coefficients)[2] + bootbeta_sd[2]), linetype = "dotted"

p3 <- ggplot(bootbeta_tb, aes(karnof90)) +
  geom_density() +
  geom_vline(xintercept = as.vector(final.fit$coefficients)[3]) +
  geom_vline(xintercept = c(as.vector(final.fit$coefficients)[3] - bootbeta_sd[3],
                            as.vector(final.fit$coefficients)[3] + bootbeta_sd[3]), linetype = "dotted"

p4 <- ggplot(bootbeta_tb, aes(karnof100)) +
  geom_density() +
  geom_vline(xintercept = as.vector(final.fit$coefficients)[4]) +
  geom_vline(xintercept = c(as.vector(final.fit$coefficients)[4] - bootbeta_sd[4],
                            as.vector(final.fit$coefficients)[4] + bootbeta_sd[4]), linetype = "dotted"

p5 <- ggplot(bootbeta_tb, aes(cd4)) +
  geom_density() +
  geom_vline(xintercept = as.vector(final.fit$coefficients)[5]) +
  geom_vline(xintercept = c(as.vector(final.fit$coefficients)[5] - bootbeta_sd[5],
                            as.vector(final.fit$coefficients)[5] + bootbeta_sd[5]), linetype = "dotted"

ggarrange(p1, p2, p3, p4, p5,
          ncol = 2,
          nrow = 3)
```



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). Standard deviation is marked by the dotted lines. 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.

### Resources

- 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)

## Gradient Boosted Trees (Madison)

### 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.

After I have a good model, I’ll produce visualizations of feature importance using SHAP values. I saw these in passing during my internship at Civis last summer, but never got to work with them directly and haven’t yet taken the time to fully understand them.

I will be using the Python `xgboost` package to give myself the added challenge of incorporating both R and Python within one RMarkdown.

My hope is that, in building model for this AIDs survival analysis task, I will come to better understand Gradient Boosting, SHAP values, and survival analysis itself.

## Cox Proportional Hazards in Generalized Boosted Models

In 2018, Tianqi Chen and other contributors added survival analysis cabability to XGBoost. Although the change is so recent that it is not yet in the XGBoost package, it exists on the github repo’s master branch for public use.

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.). This approach is nearly identical to the `CoxBoost` algorithm Binder and Schumacher developed and detail in their paper, “Allowing for mandatory covariates in boosting estimation of sparse high-dimensional survival models” (2008). Since the paper associated with XGBoost was published before their survival analysis add-on, I relied on Binder and Schumacher’s publication to give the best explanation of the Cox algorithm underying XGBoost. Their work is summarized below.

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 $\delta_i$ 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 $\beta$. We like to use a linear predictor of the form $F(x;\beta)$ where each element of the $\beta$ vector describes the influence of a single covariate. We can obtain an estimate for $\beta$ by maximizing the partial log-likelihood:

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

One way

The goal of `CoxBoost` is to estimate the parameter vector $\beta$ for a linear predictor $F(x;\beta)$ in the Cox proportional hazards model.

They 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 earlier in the context of gradient-boosting trees

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 $\beta$ 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 $\rho$ and set parameter $\tau_m$ (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. 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.

To boost a proportional hazards model, the cost function becomes the negative partial log-likelihood (Chen et al.):

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

## Analysis

See the XGBoost jupyter notebook for the analysis (can't incorporate into R yet).

## Resources

- 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