Regression in Time-Fixed Settings: Part 2

Ashley I Naimi

Spring 2022

Contents

1 Introduction to Regression for Time-to-Event Data

In the last set of notes, we looked at how to use regression to estimate risk differences, risk ratios, and odds ratios, via both conditional adjustment and marginal standardization. However, while these methods can still be used when time-to-event data are available, one may want to pursue a different course. In this set of notes, we are going to look at regression methods for time-to-event outcomes. In particular, we will focus on the Cox proportional hazards regression model, pooled logistic regression, and accelerated failure time models. We will also discuss options when competing risks are present, focusing on how to compute confounder adjusted cumulative subdistribution risks.

Throughout, we will use the time-to-event dataset we introduced in Section 1 of the course[1]:

[1] Note that I updated the simulated dataset to contain 2000 observations instead of the original 100.

```
a <- read_csv(here("data", "2022_03_09-Section1_cohort.csv"))

print(a, n = 5)
```

```
## # A tibble: 2,000 x 6
##      ID  stop exposure confounder outcome start
##   <dbl> <dbl>    <dbl>      <dbl>   <dbl> <dbl>
## 1     1  5           0          1       0     0
## 2     2  5           1          0       0     0
## 3     3  2.09        0          0       1     0
## 4     4  5           1          1       0     0
## 5     5  2.08        0          1       1     0
## # ... with 1,995 more rows
```

Recall, in these data, the outcome had three levels:

```
a %>%
    count(outcome)
```

```
## # A tibble: 3 x 2
##   outcome     n
##     <dbl> <int>
## 1       0   865
## 2       1   721
## 3       2   414
```

where `0` indicates no event, and event types `1` and `2` represent competing events. Note again that these data are considered "time-to-event" precisely because we have measured the event times measured, represented by the variable `stop`.

2 Cox Proportional Hazards Regression

Perhaps the first method that comes to mind when considering time-to-event analyses is the Cox proportional hazards (PH) model used to compute the hazard ratio. This is arguably one of the most common techniques used to evaluate time-to-event data.

The Cox model is a technique that regresses the *hazard* against a set of covariates. The hazard is a summary outcome measure like the odds, risk, or rate. It is a function of time that captures the instantaneous rate of events at a given time t, with a range of $[0, \infty]$. It is often denoted $h(t)$ or $\lambda(t)$, and is defined as:

$$\lambda(t) = \lim_{\Delta t \rightarrow 0+} \frac{P(t \leq T \leq t + \Delta t \mid t < T)}{\Delta t}$$

This function quantifies the probability that the observed event time T is between some index event time t and it's increment $t + \Delta t$, conditional on the set of individuals who are still at risk at the index event time t, all as the increment Δt for the index event time t approaches zero from the right.

Clearly, this is one of the first problems with the Cox proportional hazards regression model: **the hazard is a highly unintuitive measure of occurrence.**

More specifically, when interest lies in the effect of some exposure X, one can define the hazard as a function of the exposure as well:

$$\lambda(t \mid X = x) = \lim_{\Delta t \rightarrow 0+} \frac{P(t \leq T \leq t + \Delta t \mid t < T, X = x)}{\Delta t}$$

Now, it's quite unconventional to condition the hazard on an exposure this way. Often, one is more likely to write $\lambda(t \mid X = x)$ as $\lambda_x(t)$. But the problem with this is that one may mis-interpret this hazard as a **causal** quantity, defined via potential outcomes.[2] However, if we wanted to target the causal hazard, we have to be able to make the necessary causal identifiability assumptions.[3] Then, one can formulate the hazard in terms of potential outcomes as:

[2] Recall, subscript notation is often used to denote potential outcomes.
[3] Counterfactual consistency, no interference, exchangeabilty, positivity.

$$\lambda^x(t) = \lim_{\Delta t \to 0+} \frac{P(t \leq T^x \leq t + \Delta t \mid t < T^x)}{\Delta t}$$

where T^x represents the outcome that would be observed if the exposure X was set to some value x. If X is a binary exposure $X \in [0, 1]$,[4] then we can then define the associational hazard ratio as $\lambda(t \mid X = 1)/\lambda(t \mid X = 0)$ or the causal hazard ratio as $\lambda^{x=1}(t)/\lambda^{x=0}(t)$.

[4] Note that we use a binary exposure for simplification. This can easily be generalized to multicategory X or continuous X.

Again, for a binary exposure $X \in [0, 1]$, the Cox PH model can be motivated as follows:

$$\begin{aligned} HR =& {}^{\lambda_1(t)}/_{\lambda_0(t)} \\ \exp(\beta) =& {}^{\lambda_1(t)}/_{\lambda_0(t)} \\ \implies \lambda_1(t) =& \lambda_0(t)\exp(\beta X) \end{aligned} \qquad \text{(Cox Model)}$$

Concept Question:
Should the additive portion of the Cox model contain an intercept? Why or why not?

In effect, the Cox model defines the hazard for those with $X = 1$ as a baseline hazard $\lambda_0(t)$ multiplied by some exposure contribution $\exp(\beta X)$. In other words, the hazard for those with $X = 1$ is a multiple of the hazard for those with $X = 0$.

Why does this model imply proportional hazards? Note that the second line of the motivating equations above implies that the ratio of two hazards is $\exp(\beta)$, which does not depend on time. In other words, the ratio of two hazards cannot change over time, thus requiring constant proportion across time. An example of this with a fairly unrealistic baselien hazard is shown in Figure 1.

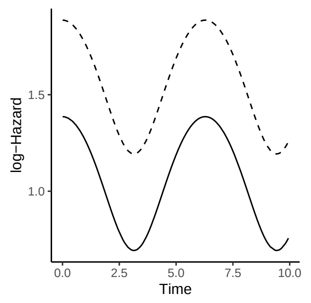

Figure 1: Illustration of the proportional hazards assumption, where the solid line represents the baseline hazard, and the dashed line represents the increase in the baseline due to some covariate.

The proportional hazards aspect of the model is often noted as an important limitation of the approach, but nonproportional hazards can easily be accommodated. Proportional hazards would be violated if, for example, the multiplicative effect of a particular variable on the baseline hazard changed over time, leading the hazards to converge or diverge (nonproportionality).

Several tests exist for evaluating proportional hazards (e.g., Schoenfeld residual plot). One easy approach is to evaluate whether there are important interactions between time and the covariates. If so, leaving these interactions in the model can be one technique to account for the nonproportional nature of the hazards.[5]

[5] If there is an interaction between time and the exposure of interest, then one has to interpret the association between the exposure and the outcome as a function of this interaction.

One property of the Cox model that is often raised as a strength of the approach is that it is *semiparametric* in that it does not assume a specific form for the baseline hazard (see Technical Note).

Technical Note:

Technically, a parametric model assumes some finite set of parameters θ. Given the parameters, predictions from the model $\hat{y}$ are independent of the observed data $\mathcal{O}$:

$$P(\hat{y} \mid \theta, \mathcal{O}) = P(\hat{y} \mid \theta).$$

That is, θ captures everything there is to know about the data. With knowledge of the parameters, we no longer need the data to obtain information about the predictions.

Because the set of parameters are finite-dimensional, the complexity of the model is bounded, even in (asymptotic, theoretical) settings where we have infinite data.

In contrast, a nonparametric model assumes θ is *infinite dimensional*. In a nonparametric model, θ is often considered a function.

Because the set of parameters are infinite-dimensional, the complexity of the model depends on the complexity of the data. Thus, even in (asymptotic, theoretical) settings where we have infinite data, we can accommodate complexity.

Finally, a semiparametric model is one that contains both finite-dimensional and infinite-dimensional parameters. The classic example is, in fact, the Cox PH model, but many other semiparametric models exist. Consider a Cox model with two covariates X_1 and X_2:

$$\lambda(t) = \lambda_0(t) \times \exp(\beta_1 X_1 + \beta_2 X_2)$$

In this model, the baseline hazard $\lambda_0(t)$ is infinite-dimensional. As a result, it can accommodate any "shape" the data may require. On the other hand, the coefficient terms $\exp(\beta_1 X_1 + \beta_2 X_2)$ are parametric. As a result, the increase in the baseline hazard due to X_1 and X_2 are encoded in this model as multiplicative. In other words, this model rules out a potential additive effect of X_1 and X_2 on the baseline hazard, for instance:

$$\lambda(t) = \lambda_0(t) + \alpha_1 X_1 + \alpha_2 X_2$$

which would not be the case if the covariate effects were specified nonparametrically.

While its semiparametric feature is indeed a strength, it does not (IMO) overcome the serious limitations inherent in the Cox model, particularly when it comes to quantifying causal effects.

One important question with the Cox model is how one can quantify model parameters, particularly with a nonparametric baseline hazard? This was one of Sir David Cox's brilliant contributions: the *partial likelihood function* (Cox, 1972, Cox (1975)). He showed using the ordered event times that the full likelihood could be re-written without incorporating the baseline hazard. For example, for K ordered event times, the partial likelihood can be written as:

$$\mathcal{L}(\beta) = \prod_{j=1}^{K} \frac{\lambda_0(T_j)\exp(\beta X_j)}{\sum_{i \in \mathcal{R}(T_j)} \lambda_0(T_j)\exp(\beta X_i)} = \prod_{j=1}^{K} \frac{\exp(\beta X_j)}{\sum_{i \in \mathcal{R}(T_j)} \exp(\beta X_i)}$$

where the index $i \in \mathcal{R}(T_j)$ refers to all individuals **at risk of the event at time** T_j. This is often referred to as the "risk-set". This has important implications for interpreting the hazard ratio, as we will see later. Moreover, because this expression requires *ordering* the event times, the presence of tied event times requires that the likelihood be modified. Many methods exist for handling ties, with Efron's method being the preferred choice in most settings (Hertz-Picciotto and Rockhill, 1997, Efron (1977)).

The Cox PH model can easily be fit in R. To do this, we'll evaluate the hazard that the outcome value is 1:

```
library(survival)

a <- a %>%
    mutate(outcome_one = as.numeric(outcome ==
        1))

cox_model <- coxph(Surv(time = start, time2 = stop,
    event = outcome_one) ~ exposure + confounder,
    data = a, ties = "efron")

summary(cox_model)
```

```
## Call:
## coxph(formula = Surv(time = start, time2 = stop, event = outcome_one) ~
##     exposure + confounder, data = a, ties = "efron")
##
##   n= 2000, number of events= 721
##
##                coef  exp(coef)  se(coef)      z Pr(>|z|)
## exposure    0.87140    2.39026   0.07692 11.329   <2e-16 ***
## confounder 0.11306   1.11970  0.07771  1.455    0.146
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##            exp(coef) exp(-coef) lower .95 upper .95
## exposure        2.39     0.4184    2.0558     2.779
## confounder      1.12     0.8931    0.9615     1.304
##
## Concordance= 0.609  (se = 0.01 )
## Likelihood ratio test= 140  on 2 df,   p=<2e-16
## Wald test            = 146.9  on 2 df,   p=<2e-16
## Score (logrank) test = 156.8  on 2 df,   p=<2e-16
```

In this particular case, we estimate an adjusted hazard ratio for the association between the exposure and the time-to-event outcome of 2.39 with 95% CIs of 2.06, 2.78.

2.1 The Problem with Cox Regression

Over a decade ago, Hernán (2010) identified important problems with the hazard ratio as a causal estimand. These were:

1) The HR can change over time. That is, the HR early on during follow-up may be different from the HR at the end of follow up. However, researchers usually report a single HR, which represents an average over the entire follow-up period.

2) The HR has a built-in selection bias. This results from the fact that the hazard is a function that conditions on surviving past time t (see also denominator of partial likelihood). So in a study with a long follow-up period, the HR may be biased by the fact that it is dominated by the presence of survivors, suggesting no exposure effect, even when the exposure is harmful early on in follow-up.

Because of these two problems, it is important to consider other estimands for quantifying exposure effects with time-to-event outcomes.

3 Pooled Logistic Regression

Pooled logistic regression is a second approach that can be used to handle time-to-event outcomes (DAGOSTINO et al., 1990). This approach is an alternative technique that can be used to approximate the hazard ratio, particularly when the outcome is rare. But it can be used for other purposes that we will see (e.g., CDF estimation) that does not require rare outcomes.

Pooled logistic regression can be implemented by fitting a standard logistic regression model for the outcome, conditional on the exposure and relevant confounders. The key condition required to fit a pooled logistic model is the way the data are constructed. So far, our dataset `a` has been analysed in the "single row per observation" format. For the pooled logistic model, the data must be transformed so that each row represents a single time interval. The choice of time interval is important, particularly if one wants to use the pooled logistic model to approximate a hazard ratio. In this case, the interval should be chosen such that the proportion of events in each interval is no more than approximately 10%.

Let's look at how we can transform the section 1 data into the single row per time-interval.

```r
# first, we create a 'stop_rounded'
# variable, which generates an integer
# count for the time on study.
a <- a %>%
    mutate(stop_rounded = ceiling(stop))  # why ceiling?

# this shows how the unrounded
# time-to-event variable relates to the
# rounded variable
a %>%
    group_by(stop_rounded) %>%
    summarize(minStop = min(stop), medianStop = median(stop),
        maxStop = max(stop))
```

```
## # A tibble: 5 x 4
##   stop_rounded minStop medianStop maxStop
##         <dbl>  <dbl>     <dbl>  <dbl>
## 1           1  0.0118    0.446  0.989
## 2           2  1.00      1.44   2.00
## 3           3  2.00      2.44   3.00
## 4           4  3.00      3.45   3.99
## 5           5  4.00      5      5
```

Once the integer time interval variable is created, we can start creating the dataset that contains a single row per time interval:

```
# create the `b` dataset, which has multiple rows per person,
# using the "uncount" function

b <- a %>%
  uncount(stop_rounded) %>% # this row expands the dataset
                            # based on the count in stop_rounded
  group_by(ID) %>%
  mutate(counter = 1,     # create a counter that can be used to sum by ID
         time_var = cumsum(counter), # sum the counter by ID:
                                     # this becomes the new "time" variable
         last_id = !duplicated(ID, fromLast = T), # flag the last row for each person
         outcome_one = outcome_one*last_id) %>% # set the new outcome
                                                # variable to 1 if last
                                                # AND old outcome is 1
  ungroup(ID) %>%
  select(ID,time_var,stop,exposure,confounder,outcome,outcome_one)

b %>% filter(ID %in% c(1,3,5,6))
```

```
## # A tibble: 16 x 7
##       ID time_var  stop exposure confounder outcome outcome_one
##    <dbl>    <dbl> <dbl>    <dbl>      <dbl>   <dbl>       <dbl>
##  1     1        1  5           0          1       0           0
##  2     1        2  5           0          1       0           0
##  3     1        3  5           0          1       0           0
##  4     1        4  5           0          1       0           0
##  5     1         5  5              0           1       0            0
##  6     3         1  2.09           0           0       1            0
##  7     3         2  2.09           0           0       1            0
##  8     3         3  2.09           0           0       1            1
##  9     5         1  2.08           0           1       1            0
## 10     5         2  2.08           0           1       1            0
## 11     5         3  2.08           0           1       1            1
## 12     6         1  4.16           1           0       1            0
## 13     6         2  4.16           1           0       1            0
## 14     6         3  4.16           1           0       1            0
## 15     6         4  4.16           1           0       1            0
## 16     6         5  4.16           1           0       1            1
```

With this dataset, we can now proceed with a pooled logistic regression model analysis. The difference between a "pooled" and standard logistic regression model is that the pooled model includes a flexible function of the time-to-event variable as a conditioning argument, in a dataset arranged in the person-time format. The pooled logistic model is often surreptitiously[6] written as:

[6] adverb; in a way that attempts to avoid notice or attention.

$$\text{logit}\{P(Y = 1 \mid T, X, C)\} = \beta_{0t} + \beta_1 X + \beta_2 C$$

where β_{0t} is a time-dependent intercept. In this model, and under the right circumstances (specifically, when the probability of the outcome in any given time interval is less than roughly 10%), one can interpret an estimate of β_1 as an approximation of the hazard ratio from the Cox PH regression model (DAGOSTINO et al., 1990). We can check this in our new dataset easily:

```
b %>%
    group_by(time_var) %>%
    summarize(meanOutcome = mean(outcome_one))
```

```
## # A tibble: 5 x 2
##   time_var meanOutcome
##      <dbl>       <dbl>
## 1        1      0.092
## 2        2      0.103
## 3         3      0.0893
## 4         4      0.0926
## 5         5      0.115
```

In practice, this time-dependent intercept can be obtained by including an indicator term for each time-to-event in the model. For example, in R, this can be accomplished using the `factor` function:

```
plr_model <- glm(outcome_one ~ factor(time_var) +
    exposure + confounder, data = b, family = binomial(link = "logit"))

round(summary(plr_model)$coefficients["exposure",
    ], 2)
```

```
##   Estimate Std. Error    z value   Pr(>|z|)
##       0.96       0.08      11.70       0.00
```

Let's compare the estimated coefficient from this model to the hazard ratio above. Recall that the Cox model yielded a HR of 2.39 with 95% CIs of 2.06, 2.78. The pooled logistic model yields an approximated HR of 2.62 with 95% confidence intervals of 2.23 and 3.08.

Note that while these confidence intervals are close to those from the Cox model, they are not technically valid.[7] In typical settings, it's customary to use the clustered sandwich variance estimator, or the clustered bootstrap.

[7] In the lab you will learn how to bootstrap the pooled logistic regression model.

Sometimes, the number of time-intervals created is too large to include in the model as a conditioning statement. For example, in a study with 60 weeks of follow-up, including weekly time-intervals with the factor function in the model would result in a model with at least 60 parameters. To deal with this, researchers will often smooth the time-intervals using polynomials or splines.

```
library(splines)

plr_model_linear <- glm(outcome_one ~ time_var +
    exposure + confounder, data = b, family = binomial(link = "logit"))

plr_model_squared <- glm(outcome_one ~ time_var +
    I(time_var^2) + exposure + confounder,
    data = b, family = binomial(link = "logit"))

plr_model_cubed <- glm(outcome_one ~ time_var +
    I(time_var^2) + I(time_var^3) + exposure +
    confounder, data = b, family = binomial(link = "logit"))

plr_model_spline <- glm(outcome_one ~ bs(time_var,
    df = 4) + exposure + confounder, data = b,
    family = binomial(link = "logit"))

round(summary(plr_model_linear)$coefficients["exposure",
    c("Estimate", "Std. Error")], 2)
```

```
##   Estimate Std. Error
##       0.96       0.08
```

```
round(summary(plr_model_squared)$coefficients["exposure",
    c("Estimate", "Std. Error")], 2)
```

```
##   Estimate Std. Error
##       0.96       0.08
```

```
round(summary(plr_model_cubed)$coefficients["exposure",
    c("Estimate", "Std. Error")], 2)
```

```
##   Estimate Std. Error
##       0.96       0.08
```

```
round(summary(plr_model_spline)$coefficients["exposure",
    c("Estimate", "Std. Error")], 2)
```

```
##   Estimate Std. Error
##       0.96       0.08
```

In these data, the functional form of the time variable does not matter.[8] This is a common usage of the pooled logistic regression model, particularly when (as we will see) the exposure and confounders are **time-varying.** One way to read the results of the pooled logistic regression output is to read the coefficient for the exposure in the model. As noted, when the outcome is rare within each unique time-interval specified in the model (or across the range of the continuous function for time in the model), this coefficient can be interpreted as an approximation to the hazard ratio.

[8] This is to be expected, because we simulated these time-to-events from an exponential distribution. This means that the simple linear term in `plr_model_linear` is actually compatible with the data generating mechanism. Additional flexibility is not required.

This is a second procedure that can be used to quantify a hazard ratio. However, one still has all the same problems with interpreting the hazard ratio. Fortunately, we can do a bit more with this model than we can with the Cox model. In particular, we can generate time-specific predicted probabilities for the outcome for each individual in the sample under exposed and unexposed states[9]:

[9] Technically, we *can* do this with the Cox model. We just need to use an estimator for the baseline hazard (which is not quantified in the Cox model), such as the Breslow estimator.

```r
plr_model_spline <- glm(outcome_one ~ bs(time_var,
    df = 4) + exposure + confounder, data = b,
    family = binomial(link = "logit"))

summary(plr_model_spline)
```

```
##
## Call:
## glm(formula = outcome_one ~ bs(time_var, df = 4) + exposure +
##     confounder, family = binomial(link = "logit"), data = b)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.6893  -0.4417  -0.3865  -0.3611   2.3521
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)           -2.69756    0.09061 -29.771   <2e-16 ***
## bs(time_var, df = 4)1  0.32591    0.38465   0.847   0.3968
## bs(time_var, df = 4)2 -0.10313    0.53658  -0.192   0.8476
## bs(time_var, df = 4)3 -0.06673    0.35062  -0.190   0.8490
## bs(time_var, df = 4)4  0.30522    0.12573   2.428   0.0152 *
## exposure               0.96225    0.08228  11.695   <2e-16 ***
## confounder             0.11403    0.08299   1.374   0.1694
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4726.6  on 7396  degrees of freedom
## Residual deviance: 4570.8  on 7390  degrees of freedom
## AIC: 4584.8
##
## Number of Fisher Scoring iterations: 5
```

```
# create three datasets, one that
# predicts under natural conditions and
# two that predict under exposure = [0,
# 1]
mu <- tibble(b, mu = predict(plr_model_spline,
    newdata = b, type = "response"))
mu1 <- tibble(b, mu1 = predict(plr_model_spline,
    newdata = transform(b, exposure = 1),
    type = "response"))
mu0 <- tibble(b, mu0 = predict(plr_model_spline,
    newdata = transform(b, exposure = 0),
    type = "response"))

# average the predictions for each
# individual stratified by time
mu <- mu %>%
    group_by(time_var) %>%
    summarize(mean_mu = mean(mu))
mu1 <- mu1 %>%
    group_by(time_var) %>%
    summarize(mean_mu1 = mean(mu1))
```

```
mu0 <- mu0 %>%
    group_by(time_var) %>%
    summarize(mean_mu0 = mean(mu0))

# cumulatively sum the predictions over
# time to estimate cumulative risk
mu <- mu %>%
    mutate(cum_risk = cumsum(mean_mu))
mu1 <- mu1 %>%
    mutate(cum_risk = cumsum(mean_mu1))
mu0 <- mu0 %>%
    mutate(cum_risk = cumsum(mean_mu0))

mu
```

```
## # A tibble: 5 x 3
##   time_var mean_mu cum_risk
##      <dbl>   <dbl>    <dbl>
## 1        1  0.0920   0.0920
## 2        2  0.103    0.195
## 3        3  0.0893   0.285
## 4        4  0.0926   0.377
## 5        5  0.115    0.493
```

```
mu1
```

```
## # A tibble: 5 x 3
##   time_var mean_mu1 cum_risk
##      <dbl>    <dbl>    <dbl>
## 1        1    0.155    0.155
## 2        2    0.175    0.330
## 3        3    0.155    0.485
## 4        4    0.161    0.646
## 5        5    0.199    0.845
```

```
mu0
```

```
## # A tibble: 5 x 3
##   time_var mean_mu0 cum_risk
##      <dbl>    <dbl>    <dbl>
## 1        1   0.0657   0.0657
## 2        2   0.0748   0.140
## 3        3   0.0653   0.206
## 4        4   0.0684   0.274
## 5        5   0.0869   0.361
```

In effect, this procedure is marginal standardization. Except, rather than only estimating the risk at the end of follow-up under exposed and unexposed states, we are quantifying the cumulative risk function over follow-up time. Note also, this cumulative risk function is marginally adjusted for the confounder. Thus, here we have a way to compute marginally standardized risk functions using a pooled logistic model. This approach, however, will only work with a single outcome of interest.

4 Multinomial Logistic Regression: Competing Risks

In the situation where competing events are present, the pooled logistic regression approach can be extended by replacing the binomial distribution with the multinomial distribution. The multinomial distribution is a generalization of the binomial distribution to scenarios where the event of interest can take on more than one value. One can define a simple multinomial logistic regression model as:

$$\text{logit}[P(Y = k \mid X, C)] = \beta_{0k} + \beta_{1k}X + \beta_{2k}C$$

for $k = 1, \dots, K$. In this simple model for a binary X variable, $\exp(\beta_1)$ can be interpreted as the following odds ratio:

$$\frac{P(Y = k \mid X = 1, C)}{P(Y = k' \mid X = 1, C)} \Big/ \frac{P(Y = k \mid X = 0, C)}{P(Y = k' \mid X = 0, C)}$$

where $Y = k'$ refers to some referent outcome level.

For example, in our example data, the outcome takes on three levels: $Y \in [0, 1, 2]$. Thus, if we take $Y = 0$ as our referent level, we obtain two odds ratios from this model:

$$\frac{P(Y = 2 \mid X = 1, C)}{P(Y = 0 \mid X = 1, C)} \bigg/ \frac{P(Y = 2 \mid X = 0, C)}{P(Y = 0 \mid X = 0, C)}$$

and

$$\frac{P(Y = 1 \mid X = 1, C)}{P(Y = 0 \mid X = 1, C)} \bigg/ \frac{P(Y = 1 \mid X = 0, C)}{P(Y = 0 \mid X = 0, C)}$$

This multinomial model can also be adapted to the time-to-event setting in the same way we adapted the logistic regression model. In effect, we can fit a pooled multinomial model as:

$$\text{logit}[P(Y = k \mid X, C)] = \beta_{0tk} + \beta_{1k}X + \beta_{2k}C$$

which effectively makes each outcome-specific intercept a function of time.

In R, their are several packages that can be used to fit a multinomial model, including the `multinom` function in the `nnet`, and the `vglm` function in the `VGAM` package. We'll use the latter here.

Let's first look at the data we'll be fitting:

```r
b %>%
    filter(ID %in% c(1, 3, 5, 6))
```

```
## # A tibble: 16 x 7
##       ID time_var  stop exposure confounder outcome outcome_one
##    <dbl>    <dbl> <dbl>    <dbl>      <dbl>   <dbl>       <dbl>
##  1     1        1  5           0          1       0           0
##  2     1        2  5           0          1       0           0
##  3     1        3  5           0          1       0           0
##  4     1        4  5           0          1       0           0
##  5     1        5  5           0          1       0           0
##  6     3        1  2.09        0          0       1           0
##  7     3        2  2.09        0          0       1           0
##  8     3        3  2.09        0          0       1           1
##  9     5        1  2.08        0          1       1           0
## 10     5        2  2.08        0           1       1            0
## 11     5        3  2.08        0           1       1            1
## 12     6        1  4.16        1           0       1            0
## 13     6        2  4.16        1           0       1            0
## 14     6        3  4.16        1           0       1            0
## 15     6        4  4.16        1           0       1            0
## 16     6        5  4.16        1           0       1            1
```

Note that the `outcome` variable is indeed a three-level variable:

```
b %>%
    count(outcome)
```

```
## # A tibble: 3 x 2
##   outcome     n
##     <dbl> <int>
## 1       0  4325
## 2       1  1972
## 3       2  1100
```

To implement the pooled multinomial model for competing risks, we can use the `vglm` function. To allow for flexibility along the time dimension, we'll use splines:

```
library(VGAM)
```

```
## Loading required package: stats4

##
## Attaching package: 'VGAM'

## The following object is masked from 'package:tidyr':
##
##     fill
```

```
pmr1 <- vglm(outcome ~ time_var + exposure +
    confounder, data = b, family = multinomial(refLevel = 1))  ## note the need for refLevel

summary(pmr1)@coef3[c("exposure:1", "exposure:2"),
    ]
```

```
##              Estimate Std. Error   z value     Pr(>|z|)
## exposure:1  0.9154848 0.06117863  14.96413 1.259647e-50
## exposure:2 -1.6044225 0.13028654 -12.31457 7.561861e-35
```

However, once again, the coefficients in this model will be challenging to interpret. We can implement the same procedure with the binomial model to obtain adjusted sub-distribution risk curves. To do this, it's useful to see what we obtain when we apply the predict function to an object resulting from the fit of the `vglm` function:

```
head(predict(pmr1, newdata = b, type = "response"))
```

```
##           0         1          2
## 1 0.4011168 0.2889056 0.30997764
## 2 0.5091684 0.2419770 0.24885458
## 3 0.6162640 0.1932444 0.19049153
## 4 0.7130646 0.1475355 0.13939987
## 5 0.7935504 0.1083352 0.09811438
## 6 0.3776483 0.5767495 0.04560218
```

The output from the predict function with a multinomial model (fitted with `vglm`) is a K column dataset, where K is the number of outcome levels. We thus have to adjust our code to ensure we select the correct probabilites when creating standardized risk curves:

```
pmr1 <- vglm(outcome ~ scale(time_var) +
    exposure + confounder, data = b, family = multinomial(refLevel = 1))

# create three datasets, one that
```

```r
# predicts under natural conditions and
# two that predict under exposure = [0,
# 1] FOR OUTCOME = 1
mu_1 <- tibble(b, mu_1 = predict(pmr1, newdata = b,
    type = "response")[, 2])
mu_11 <- tibble(b, mu_11 = predict(pmr1,
    newdata = transform(b, exposure = 1),
    type = "response")[, 2])
mu_10 <- tibble(b, mu_10 = predict(pmr1,
    newdata = transform(b, exposure = 0),
    type = "response")[, 2])
## FOR OUTCOME = 2
mu_2 <- tibble(b, mu_2 = predict(pmr1, newdata = b,
    type = "response")[, 3])
mu_21 <- tibble(b, mu_21 = predict(pmr1,
    newdata = transform(b, exposure = 1),
    type = "response")[, 3])
mu_20 <- tibble(b, mu_20 = predict(pmr1,
    newdata = transform(b, exposure = 0),
    type = "response")[, 3])

# average the predictions for each
# individual stratified by time FOR
# OUTCOME = 1
mu_1 <- mu_1 %>%
    group_by(time_var) %>%
    summarize(mean_mu_1 = mean(mu_1))
mu_11 <- mu_11 %>%
    group_by(time_var) %>%
    summarize(mean_mu_11 = mean(mu_11))
mu_10 <- mu_10 %>%
    group_by(time_var) %>%
    summarize(mean_mu_10 = mean(mu_10))
## FOR OUTCOME = 2
```

```
mu_2 <- mu_2 %>%
    group_by(time_var) %>%
    summarize(mean_mu_2 = mean(mu_2))
mu_21 <- mu_21 %>%
    group_by(time_var) %>%
    summarize(mean_mu_21 = mean(mu_21))
mu_20 <- mu_20 %>%
    group_by(time_var) %>%
    summarize(mean_mu_20 = mean(mu_20))

# cumulatively sum the predictions over
# time to estimate cumulative risk FOR
# OUTCOME = 1
mu_1 <- mu_1 %>%
    mutate(cum_risk = cumsum(mean_mu_1))
mu_11 <- mu_11 %>%
    mutate(cum_risk = cumsum(mean_mu_11))
mu_10 <- mu_10 %>%
    mutate(cum_risk = cumsum(mean_mu_10))
## FOR OUTCOME = 2
mu_2 <- mu_2 %>%
    mutate(cum_risk = cumsum(mean_mu_2))
mu_21 <- mu_21 %>%
    mutate(cum_risk = cumsum(mean_mu_21))
mu_20 <- mu_20 %>%
    mutate(cum_risk = cumsum(mean_mu_20))

mu_1
```

```
## # A tibble: 5 x 3
##   time_var mean_mu_1 cum_risk
##      <dbl>     <dbl>    <dbl>
## 1        1     0.371    0.371
## 2        2     0.307    0.678
## 3        3     0.241    0.919
## 4        4     0.183    1.10
## 5        5     0.133    1.24
```

```
mu_11
```

```
## # A tibble: 5 x 3
##   time_var mean_mu_11 cum_risk
##      <dbl>      <dbl>    <dbl>
## 1        1      0.588    0.588
## 2        2      0.496    1.08
## 3        3      0.401    1.49
## 4        4      0.310    1.80
## 5        5      0.231    2.03
```

```
mu_10
```

```
## # A tibble: 5 x 3
##   time_var mean_mu_10 cum_risk
##      <dbl>      <dbl>    <dbl>
## 1        1     0.281     0.281
## 2        2     0.232     0.513
## 3        3     0.182     0.695
## 4        4     0.138     0.833
## 5        5     0.0999    0.933
```

```
mu_2
```

```
## # A tibble: 5 x 3
##   time_var mean_mu_2 cum_risk
##      <dbl>     <dbl>    <dbl>
## 1        1    0.215     0.215
## 2        2    0.172     0.387
## 3        3    0.132     0.519
## 4        4    0.0963    0.615
## 5        5    0.0676    0.683
```

```
mu_21
```

```
## # A tibble: 5 x 3
##   time_var mean_mu_21 cum_risk
##      <dbl>      <dbl>    <dbl>
## 1        1     0.0481   0.0481
## 2        2     0.0389   0.0870
## 3        3     0.0301   0.117
## 4        4     0.0223   0.139
## 5        5     0.0159   0.155
```

```
mu_20
```

```
## # A tibble: 5 x 3
##   time_var mean_mu_20 cum_risk
##      <dbl>      <dbl>    <dbl>
## 1        1     0.285     0.285
## 2        2     0.226     0.511
## 3        3     0.170     0.681
## 4        4     0.123     0.805
## 5        5     0.0857    0.890
```

References

D. R. Cox. Regression models and life-tables. *J R Stat Soc B*, 34(2):187–220, 1972.

D. R. Cox. Partial likelihood. *Biometrika*, 62(2):269–276, 1975.

R. B. DAGOSTINO, M. L. LEE, A. J. BELANGER, L. A. CUPPLES, K. ANDERSON, and W. B. KANNEL. Relation of pooled logistic-regression to time-dependent cox regression-analysis - the framingham heart-study. *STATISTICS IN MEDICINE*, 9(12):1501–1515, Dec 1990. ISSN 0277-6715. DOI: 10.1002/sim.4780091214.

Bradley Efron. The efficiency of cox's likelihood function for censored data. *J Am Stat Assoc*, 72(359):557—565, 1977.

M. A. Hernán. The hazards of hazard ratios. *Epidemiol*, 21:13–5, 2010.

Irva Hertz-Picciotto and Beverly Rockhill. Validity and efficiency of approximation methods for tied survival times in cox regression. *Biometrics*, 53(3): 1151–1156, 1997.