DEM 7223 - Event History Analysis - Discrete Time Hazard Model 2

Corey S. Sparks, PhD

03 November, 2022

Notes

As we saw last week, when we fit the discrete time hazards model to the person-period data, we fit the model:

$$logit(h(t)) = \alpha_j + x'\beta$$

Where the α_j's represent the logit-hazard for the distinct risk periods, and the β's represent the effects of covariates on the hazard.

This formulation is called the "general" form because it allows the shape of the hazard function to vary at each time point, as defined by the α_j's!

This allows great flexibility in modeling the changing nature of risk

This model specification by definition sets no constraints on the shape of the baseline function because we estimate the hazard at every time point.

This specification is:

- Easily interpretable in multiple scales
- Informative about the shape of the hazard function
- Consistent with estimates from life tables
  - If we include no predictors, we basically estimate the life table $q(x)$

Alternative time specifications

We can specify alternative ways of modeling the hazard function

Instead of having one α_j for each time point, we may be able to capture the shape of the hazard with fewer parameters.

Because the general model is not parsimonious, the estimates of the baseline hazard can fluctuate dramatically from time to time owing solely to sampling variation

Also, if there are not many failures at particular times, we can have low certainty about the parameter estimate

Although time in the general model is treated discretely, we can specify continuous time models as well by specifying time as a continuous, rather than a categorical predictor in our model

This is done by assuming time is either:

- Constant – hazard is constant at all time, time is not in the model at all
- Linear – hazard increases/decreases with time
- Quadratic – hazard is a nonlinear curve over time with 1 inflection point
- Cubic – hazard is nonlinear over time with 2 inflection points
- Higher order – hazard is nonlinear over time with >2 inflection points
- Spline basis – nonlinear but not specifically using polynomials

Each of these models specifies a more parsimonious version of the general form, with fewer parameters, *if the number of time points is large*

But, they will never fit the data as well as the general model

Comparing model fit for different time specifications

To compare discrete time models we use standard model comparison methods

- Model Deviance = (-2*log likelihood)
- Model AIC = (-2 log likelihood + 2*(p+1))

For example, if we fit the constant hazard, the linear hazard, a cubic hazard and a quartic hazard, we compare the performance of each model to the performance of the general model (with 1 parameter for each time point)

We get:

We see the General model is the best fitting, but the quartic model is the next best.

Model Type	Deviance	Deviance Difference from General	AIC	AIC difference from General	# parameters
constant	30172.94	2675.54	30174.94	2665.54	1
linear	27566.57	69.17	27570.57	61.17	2
square	27551.18	53.78	27557.18	47.78	3
cubic	27550.39	52.99	27558.39	**48.99**	4
quartic	27500.44	3.04	27510.44	***1.04***	5
general	27497.4	0	27509.4	0	6

Figure 1: AIC Table

Constructing Survival Curves for the discrete time model

We have seen how to plot the hazard function for the discrete time model, but we can also plot the survival function estimates as well.

If we use the product-limit estimator, then we can estimate survival to time t as:

$$S(t) = S_{t-1} * (1 - h_t)$$

Examples

This example will illustrate how to fit the discrete time hazard model to person-period data. Specifically, this example illustrates various parameterizartions of time in the discrete time model. In this example, I will use the event of a couple having a second birth. The data for this example come from the data Demographic and Health Survey for 2012 children's recode file. This file contains information for all births in the last 5 years prior to the survey.

The second example will use data from the IPUMS NHIS and examine alternative methods for coding time between survey and mortality follow up.

Continuous duration outcome - DHS data

load the data

```
library(tidyverse)
```

```
-- Attaching packages --------------------------------------- tidyverse 1.3.2 --
v ggplot2 3.3.6      v purrr   0.3.5
v tibble  3.1.8      v dplyr   1.0.10
v tidyr   1.2.1      v stringr 1.4.1
v readr   2.1.3      v forcats 0.5.2
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
```

```
library(haven)
#load the data
dat<-read_dta("../data/ZAIR71FL.DTA")
dat<-zap_labels(dat)
```

Event - Second birth occurrence

In the DHS individual recode file, information on every live birth is collected using a retrospective birth history survey mechanism.

Since our outcome is time between first and second birth, we must select as our risk set, only women who have had a first birth.

The `bidx` variable indexes the birth history and if `bidx_01` is not missing, then the woman should be at risk of having a second birth (i.e. she has had a first birth, i.e. `bidx_01==1`).

I also select only non-twin births (`b0 == 0`).

The DHS provides the dates of when each child was born in Century Month Codes.

To get the interval for women who *actually had* a second birth, that is the difference between the CMC for the first birth `b3_01` and the second birth `b3_02`, but for women who had not had a second birth by the time of the interview, the censored time between births is the difference between `b3_01` and `v008`, the date of the interview.

We have 6124 women who are at risk of a second birth.

```
sub<-dat %>%
  filter(bidx_01==1&b0_01==0)%>%
  transmute(CASEID=caseid,
                  int.cmc=v008,
                  fbir.cmc=b3_01,
                  sbir.cmc=b3_02,
                  marr.cmc=v509,
                  rural=v025,
                  educ=v106,
                  age = v012,
                  agec=cut(v012, breaks = seq(15,50,5), include.lowest=T),
                  partneredu=v701,
                  partnerage=v730,
                  weight=v005/1000000,
                  psu=v021,
                  strata=v022)%>%
  select(CASEID, int.cmc, fbir.cmc, sbir.cmc, marr.cmc, rural, educ, age, agec, partneredu
  mutate(agefb = (age - (int.cmc - fbir.cmc)/12))%>%
  mutate(secbi = ifelse(is.na(sbir.cmc)==T,
                    int.cmc - fbir.cmc,
                    fbir.cmc - sbir.cmc),
         b2event = ifelse(is.na(sbir.cmc)==T,0,1))
```

Create the person-period file

The distinction between the way we have been doing things and the discrete time model, is that we treat time discretely, versus continuously. This means that we transform the data from the case-duration data format to the person-period format. For this example, a natural choice would be year, since we have intervals of equal length (12 months each).

R provides a useful function called `survSplit()` in the `survival` library that will split a continuous duration into discrete periods.

```
library(survey)
```

```
Loading required package: grid
```

```
Loading required package: Matrix
```

```

Attaching package: 'Matrix'
```

```
The following objects are masked from 'package:tidyr':

    expand, pack, unpack

Loading required package: survival


Attaching package: 'survey'

The following object is masked from 'package:graphics':

    dotchart
```

```r
library(survival)
pp<-survSplit(Surv(secbi, b2event)~. ,
              data = sub[sub$secbi>0,],
              cut=seq(0,240, 12),
              episode="year_birth")

pp$year <- pp$year_birth-1
pp<-pp[order(pp$CASEID, pp$year_birth),]
knitr::kable(head(pp[, c("CASEID", "secbi", "b2event", "year", "educ", "agefb")], n=20))
```

CASEID	secbi	b2event	year	educ	agefb
1 13 2	12	0	1	1	24.66667
1 13 2	24	0	2	1	24.66667
1 13 2	36	0	3	1	24.66667
1 13 2	48	0	4	1	24.66667
1 13 2	60	0	5	1	24.66667
1 13 2	72	0	6	1	24.66667
1 13 2	84	0	7	1	24.66667
1 13 2	96	0	8	1	24.66667
1 13 2	108	0	9	1	24.66667
1 13 2	120	0	10	1	24.66667
1 13 2	132	0	11	1	24.66667
1 13 2	144	0	12	1	24.66667
1 13 2	156	0	13	1	24.66667
1 13 2	168	0	14	1	24.66667
1 13 2	180	0	15	1	24.66667
1 13 2	192	0	16	1	24.66667

CASEID	secbi	b2event	year	educ	agefb
1 13 2	196	0	17	1	24.66667
1 36 2	12	0	1	2	22.75000
1 36 2	24	0	2	2	22.75000
1 36 2	36	0	3	2	22.75000

We see that each woman is not in the data for multiple "risk periods", until they experience the event (birth) or are censored.

Discrete time model

So, the best thing about the discrete time model, is that it's just logistic regression. Each risk period is treated as a single Bernoulli trial, and the birth can either occur (y=1) or not (y=0) in the period. This is how we get the hazard of the event, as the estimated probability of failure in each discrete time period. So, any method you would like to use to model this probability would probably work (logit, probit models), but I will show two standard approaches.

First, we will use the traditional logit link to the binomial distribution, then we will use the complementary log-log link. The latter is used because it preserves the proportional hazards property of the model, as in the Cox model.

```
options(survey.lonely.psu = "adjust")
des<-survey::svydesign(ids=~psu,
                strata=~strata,
                data=pp,
                weight=~weight )
```

```
#Fit the basic logistic model with ONLY time in the model
#I do -1 so that no intercept is fit in the model, and we get a hazard estimate for each t
fit.0<-svyglm(b2event~as.factor(year)-1,
              design=des,
              family=binomial(link="cloglog"))
```

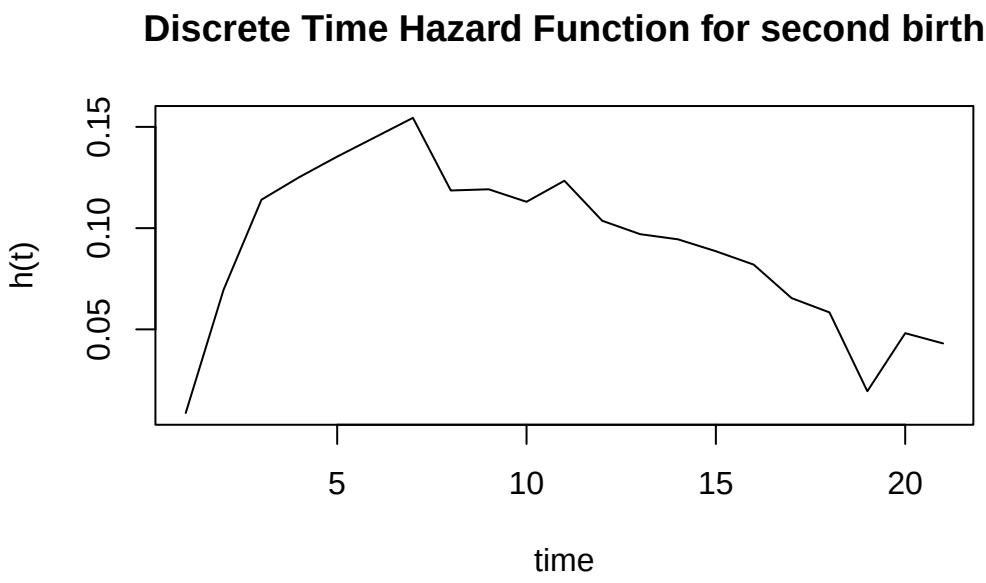
```
Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```
summary(fit.0)
```

```
Call:
svyglm(formula = b2event ~ as.factor(year) - 1, design = des,
    family = binomial(link = "cloglog"))

Survey design:
survey::svydesign(ids = ~psu, strata = ~strata, data = pp, weight = ~weight)

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
as.factor(year)1  -4.73285    0.16969 -27.891  < 2e-16 ***
as.factor(year)2  -2.59301    0.06627 -39.126  < 2e-16 ***
as.factor(year)3  -2.05019    0.05698 -35.982  < 2e-16 ***
as.factor(year)4  -1.94422    0.05018 -38.746  < 2e-16 ***
as.factor(year)5  -1.85472    0.05608 -33.072  < 2e-16 ***
as.factor(year)6  -1.77539    0.05706 -31.115  < 2e-16 ***
as.factor(year)7  -1.69999    0.06850 -24.819  < 2e-16 ***
as.factor(year)8  -2.00585    0.08490 -23.625  < 2e-16 ***
as.factor(year)9  -2.00025    0.09480 -21.099  < 2e-16 ***
as.factor(year)10 -2.06047    0.11273 -18.278  < 2e-16 ***
as.factor(year)11 -1.96067    0.11481 -17.078  < 2e-16 ***
as.factor(year)12 -2.15754    0.13221 -16.319  < 2e-16 ***
as.factor(year)13 -2.23079    0.17452 -12.782  < 2e-16 ***
as.factor(year)14 -2.26038    0.17878 -12.644  < 2e-16 ***
as.factor(year)15 -2.33106    0.20768 -11.225  < 2e-16 ***
as.factor(year)16 -2.41583    0.19583 -12.336  < 2e-16 ***
as.factor(year)17 -2.65901    0.31435  -8.459  < 2e-16 ***
as.factor(year)18 -2.78069    0.34502  -8.060 3.54e-15 ***
as.factor(year)19 -3.92030    0.47573  -8.241 9.06e-16 ***
as.factor(year)20 -2.98481    0.38224  -7.809 2.24e-14 ***
as.factor(year)21 -3.10096    0.51659  -6.003 3.18e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1.000026)

Number of Fisher Scoring iterations: 7
```

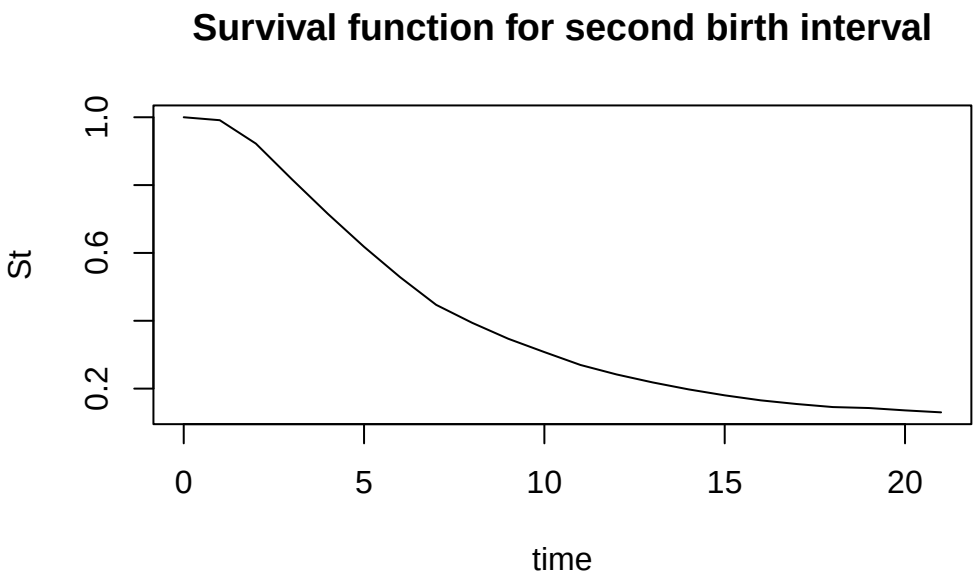
```r
#Plot the hazard function on the probability scale
haz<-1/(1+exp(-coef(fit.0)))
time<-seq(1,21,1)
```

```
plot(haz~time, type="l", ylab="h(t)")
title(main="Discrete Time Hazard Function for second birth")
```



Plot the survival function estimate

```
St<-NA
time<-1:length(haz)
St[1]<-1-haz[1]
for(i in 2:length(haz)){
  St[i]<-St[i-1]* (1-haz[i])
}


St<-c(1, St)
time<-c(0, time)
plot(y=St,x=time, type="l",
     main="Survival function for second birth interval")
```



Alternative time specifications

Here, we can specify how we want time in our model. The general model fits a hazard at every time point. If you have a lot of time points, and especially if you have low numbers of events at some time points, this can be computationally expensive.

We can, however, specify time as a linear or quadratic term, and the model will not fit separate hazards at all times, instead, the baseline hazard will be a linear or curvilinear function of time.

```
#Linear term for time
fit.0<-svyglm(b2event~1,
              design=des ,
              family=binomial(link="cloglog"))
```

```
Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```
summary(fit.0)
```

```
Call:
svyglm(formula = b2event ~ 1, design = des, family = binomial(link = "cloglog"))

Survey design:
survey::svydesign(ids = ~psu, strata = ~strata, data = pp, weight = ~weight)

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -2.22706    0.01918  -116.1   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1.000026)

Number of Fisher Scoring iterations: 5
```

```r
1/(1+exp(-coef(fit.0)))
```

```
(Intercept)
 0.09734632
```

```r
#Linear term for time
fit.l<-svyglm(b2event~year,
              design=des ,
              family=binomial(link="cloglog"))
```

```
Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```r
summary(fit.l)
```

```
Call:
svyglm(formula = b2event ~ year, design = des, family = binomial(link = "cloglog"))

Survey design:
survey::svydesign(ids = ~psu, strata = ~strata, data = pp, weight = ~weight)

Coefficients:
```

```
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -2.416650   0.028047 -86.164  < 2e-16 ***
year         0.033726   0.004761   7.084 3.47e-12 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 0.9952062)

Number of Fisher Scoring iterations: 5
```

Which shows the hazard increases over time, which makes a lot of sense in the context of this outcome. Now we can consider quadratic terms for time and a natural spline function of time as well:

```
fit.s<-svyglm(b2event~year+I(year^2),
              design=des ,
              family=binomial(link="cloglog"))
```

```
Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```
summary(fit.s)
```

```
Call:
svyglm(formula = b2event ~ year + I(year^2), design = des, family = binomial(link = "cloglog"

Survey design:
survey::svydesign(ids = ~psu, strata = ~strata, data = pp, weight = ~weight)

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -3.35596    0.06434  -52.16   <2e-16 ***
year         0.38369    0.02195   17.48   <2e-16 ***
I(year^2)   -0.02209    0.00149  -14.82   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1.009634)

Number of Fisher Scoring iterations: 6
```

```
fit.c<-svyglm(b2event~year+I(year^2)+I(year^3 ),
              design=des ,
              family=binomial(link="cloglog"))
```

```
Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```
summary(fit.c)
```

```
Call:
svyglm(formula = b2event ~ year + I(year^2) + I(year^3), design = des,
    family = binomial(link = "cloglog"))

Survey design:
survey::svydesign(ids = ~psu, strata = ~strata, data = pp, weight = ~weight)

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -4.1368559  0.0912631  -45.33   <2e-16 ***
year         0.8285081  0.0401488   20.64   <2e-16 ***
I(year^2)   -0.0843188  0.0048356  -17.44   <2e-16 ***
I(year^3)    0.0023341  0.0001608   14.52   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 0.964793)

Number of Fisher Scoring iterations: 6
```

```
fit.q<-svyglm(b2event~year+I(year^2)+I(year^3 )+I(year^4),
              design=des ,
              family=binomial(link="cloglog"))
```

```
Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```
summary(fit.q)
```

```
Call:
svyglm(formula = b2event ~ year + I(year^2) + I(year^3) + I(year^4),
    design = des, family = binomial(link = "cloglog"))

Survey design:
survey::svydesign(ids = ~psu, strata = ~strata, data = pp, weight = ~weight)

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -5.216e+00  1.431e-01 -36.444  < 2e-16 ***
year         1.659e+00  1.006e-01  16.486  < 2e-16 ***
I(year^2)   -2.680e-01  2.281e-02 -11.750  < 2e-16 ***
I(year^3)    1.713e-02  1.943e-03   8.816  < 2e-16 ***
I(year^4)   -3.835e-04  5.415e-05  -7.082 3.53e-12 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 0.9679146)

Number of Fisher Scoring iterations: 6
```

```r
#Spline
library(splines)
fit.sp<-svyglm(b2event~ns(year, df = 3),
               design=des ,
               family=binomial(link="cloglog"))
```

```
Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```r
summary(fit.sp)
```

```
Call:
svyglm(formula = b2event ~ ns(year, df = 3), design = des, family = binomial(link = "cloglog"

Survey design:
survey::svydesign(ids = ~psu, strata = ~strata, data = pp, weight = ~weight)

Coefficients:
```

```
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)       -3.82491    0.07575 -50.492  < 2e-16 ***
ns(year, df = 3)1  0.95545    0.11930   8.009 4.97e-15 ***
ns(year, df = 3)2  3.18995    0.16248  19.633  < 2e-16 ***
ns(year, df = 3)3 -0.16044    0.15785  -1.016     0.31
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 0.9607684)

Number of Fisher Scoring iterations: 6
```

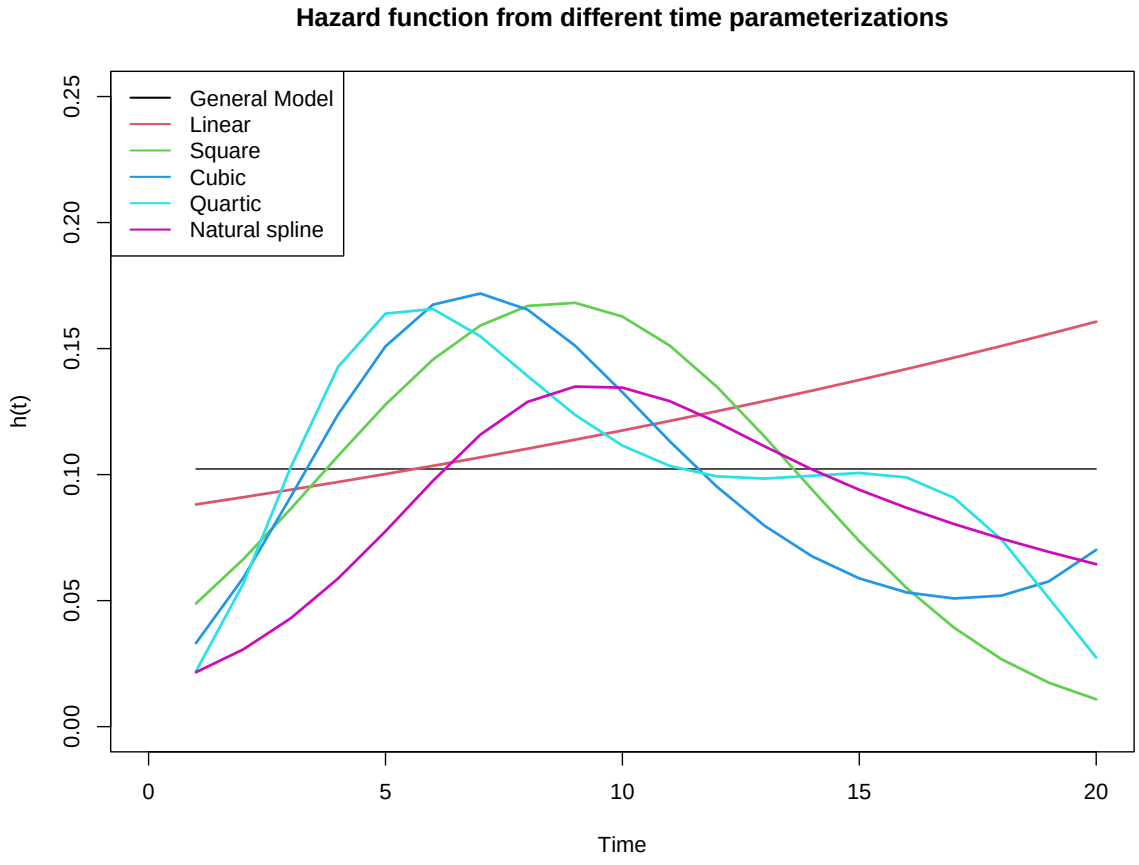
Now, let's look at the hazards:

```
dat<-expand.grid(year=seq(1,20,1))
dat$genmod<-predict(fit.0, newdata=data.frame(year=as.factor(1:20 )), type="response")
dat$lin<-predict(fit.l, newdata=dat, type="response")
dat$sq<-predict(fit.s, newdata=dat, type="response")
dat$cub<-predict(fit.c, newdata=dat, type="response")
dat$quart<-predict(fit.q, newdata=dat, type="response")
dat$spline<-predict(fit.sp, newdata=expand.grid(year=seq(1,20,1)), type="response")

dat
```

```
   year    genmod        lin         sq        cub      quart     spline
1     1 0.1022329 0.08815054 0.04883565 0.03313604 0.02193840 0.02158409
2     2 0.1022329 0.09103227 0.06646075 0.05908727 0.05678264 0.03066431
3     3 0.1022329 0.09400326 0.08641637 0.09120755 0.10291220 0.04298980
4     4 0.1022329 0.09706594 0.10742423 0.12393847 0.14275615 0.05878068
5     5 0.1022329 0.10022279 0.12778480 0.15089228 0.16388223 0.07750134
6     6 0.1022329 0.10347630 0.14561261 0.16740835 0.16570870 0.09748243
7     7 0.1022329 0.10682904 0.15912493 0.17185330 0.15489811 0.11581018
8     8 0.1022329 0.11028360 0.16691445 0.16543487 0.13909384 0.12887238
9     9 0.1022329 0.11384259 0.16815444 0.15116988 0.12371293 0.13492314
10   10 0.1022329 0.11750868 0.16271246 0.13267571 0.11152262 0.13449755
11   11 0.1022329 0.12128457 0.15116809 0.11320260 0.10342332 0.12912848
12   12 0.1022329 0.12517297 0.13473441 0.09509968 0.09928661 0.12074750
13   13 0.1022329 0.12917666 0.11508504 0.07971626 0.09840487 0.11121754
14   14 0.1022329 0.13329839 0.09410183 0.06759968 0.09953025 0.10206049
15   15 0.1022329 0.13754099 0.07358435 0.05881197 0.10065714 0.09396874
16   16 0.1022329 0.14190726 0.05498699 0.05323979 0.09886525 0.08680359
17   17 0.1022329 0.14640005 0.03924959 0.05085754 0.09078436 0.08039563
```

```
18   18 0.1022329 0.15102220 0.02675798 0.05198023 0.07428135 0.07460558
19   19 0.1022329 0.15577656 0.01742441 0.05761491 0.05103379 0.06931880
20   20 0.1022329 0.16066599 0.01084067 0.07013976 0.02742312 0.06444111
```

```
plot(genmod~year, dat, type="l", ylab="h(t)", xlab="Time", ylim=c(0, .25), xlim=c(0, 20))
title(main="Hazard function from different time parameterizations")
lines(lin~year, dat, col=2, lwd=2)
lines(sq~year, dat, col=3, lwd=2)
lines(cub~year, dat, col=4, lwd=2)
lines(quart~year, dat, col=5, lwd=2)
lines(spline~year, dat, col=6, lwd=2)

legend("topleft",
       legend=c("General Model", "Linear","Square", "Cubic", "Quartic", "Natural spline"),
       col=1:6, lwd=1.5)
```



```r
#AIC table
aic<-round(c(
  fit.l$deviance+2*length(fit.l$coefficients),
  fit.s$deviance+2*length(fit.s$coefficients),
  fit.c$deviance+2*length(fit.c$coefficients),
  fit.q$deviance+2*length(fit.q$coefficients),
  fit.sp$deviance+2*length(fit.sp$coefficients),
  fit.0$deviance+2*length(fit.0$coefficients)),2)

#compare all aics to the one from the general model
dif.aic<-round(aic-aic[6],2)
data.frame(model =c( "linear","square", "cubic", "quartic","spline", "general"),
           aic=aic,
           aic_dif=dif.aic)
```

```
    model      aic  aic_dif
1  linear 25090.01   -90.04
2  square 24386.34  -793.71
3   cubic 24158.01 -1022.04
4 quartic 24010.14 -1169.91
5  spline 24002.84 -1177.21
6 general 25180.05     0.00
```

So the general model, quartic and spline give the same AIC points, but the square and cubic models also fit equally as well.

IPUMS NHIS Mortality example

```r
library(survey)
library(survival)
library(car)
```

```
Loading required package: carData
```

```
Attaching package: 'car'
```

```
The following object is masked from 'package:dplyr':

    recode
```

```
The following object is masked from 'package:purrr':

    some
```

```
dat<-haven::read_dta("C:/Users/ozd504/OneDrive - University of Texas at San Antonio/classe
names(dat)
```

```
 [1] "year"         "serial"       "strata"       "psu"          "nhishid"
 [6] "hhweight"     "region"       "pernum"       "nhispid"      "hhx"
[11] "fmx"          "px"           "perweight"    "sampweight"   "fweight"
[16] "supp2wt"      "intervwyr"    "astatflg"     "cstatflg"     "telephone"
[21] "age"          "sex"          "marstat"      "marst"        "marstcohab"
[26] "cohabmarst"   "cohabevmar"   "birthyr"      "relate"       "famsize"
[31] "momed"        "racea"        "hispeth"      "yrsinus"      "hispyn"
[36] "usborn"       "citizen"      "racenew"      "educrec2"     "empstat"
[41] "pooryn"       "incfamr82on"  "gotnewelf"    "gotssiwhy"    "gotnonssdis"
[46] "gotdiv"       "gotchsup"     "gotstamp"     "poverty"      "poverty2"
[51] "ownership"    "lowrent"      "health"       "height"       "weight"
[56] "bmicalc"      "bmi"          "bedayr"       "hstatyr"      "wldayr"
[61] "usualpl"      "typplsick"    "routcare"     "delaycost"    "famdelaycono"
[66] "famdelaycost" "famybarcar"   "famybarcarno" "placecar"     "delayappt"
[71] "delayhrs"     "delayphone"   "delaytrans"   "delaywait"    "ybarcare"
[76] "ybardental"   "ybarglass"    "ybarmeds"     "ybarmental"   "alc5upyr"
[81] "smokev"       "smokagereg"   "mortelig"     "mortstat"     "mortdody"
[86] "mortucodld"   "mortwt"       "mortcms"      "mortndi"      "mortssa"
[91] "mortwtsa"
```

```
sub<-subset(dat, dat$mortelig==1&is.na(dat$racenew)==F)
samps<-sample(1:length(sub$year), size = 100000, replace = F)
sub<-sub[samps,]
sub<-zap_labels(sub)
```

alternative codings of failure time

In mortality analysis, but in any type of case-duration data, we can construct our event indicator to represent if an event occurs during a set risk period. In the NHIS mortality, we could code the death as a few different ways:

If you are interested in the *Age at death*, then you will measure your outcome as the age when they died

Coding age at death. If the person died, then it is equal to the year the person died, minus the year in which they were born. If the person was still alive, then it is the difference between 2009 (year of follow up in this data file)

```
sub$d.age<-ifelse(sub$mortstat==1,sub$mortdody-(sub$year-sub$age) ,
                  ifelse(sub$mortstat==2,2009-(sub$year-sub$age), NA))
```

If you are interested in the risk of death in a follow up period, then you could code it as: - Did the person die in the follow up period at all? No time specified - Did the person die in a 1, 5, or 10 year period? This specifies a fixed risk period

```
sub$d.event<-ifelse(sub$mortstat==1,1,0) #did they die?

sub$timetodeath<-ifelse(sub$mortstat ==1,
                        sub$mortdody-sub$year  ,
                        2009 - sub$year ) #how long after the interview did they die?

sub$d1yr<-ifelse(sub$timetodeath<=1&sub$mortstat==1, 1,0) #did they die within 1 year of t
sub$d5yr<-ifelse(sub$timetodeath<=5&sub$mortstat==1, 1,0) #did they die within 5 years of
```

Other variables

```
library(car)
sub$married<-Recode(sub$marstat,
                    recodes="00=NA; 10:13='married'; 20:40='sep'; 50='nm'; 99=NA" ,
                    as.factor=T )
sub$male<-ifelse(sub$sex==1,1,0)
sub$mwt<-sub$mortwt/mean(sub$mortwt, na.rm=T)

sub$age5<-cut(sub$age,seq(15,85, 5))

sub$race<-Recode(sub$racenew,
                 recodes ="10='wht'; 20 ='blk'; 30:61='other'; 97:99=NA",
                 as.factor=T)
sub$college<-Recode(sub$educrec2,
                    recodes="00=NA; 10:42='hs or less'; 50:53='some coll'; 54:60='coll'; e
                    as.factor=T)
sub$black<-ifelse(sub$race=='blk',1,0)
sub$oth<-ifelse(sub$race=='other',1,0)
sub$hs<-ifelse(sub$college=='hs or less',1,0)
sub$col1<-ifelse(sub$college=='some coll',1,0)
sub$sep<-ifelse(sub$married=='sep',1,0)
```

```
sub$nm<-ifelse(sub$married=='nm',1,0)

sub$race<-Recode(sub$racenew, recodes ="10='wht'; 20 ='blk'; 30:61='other'; 97:99=NA", as.
sub$hisp<-Recode(sub$hispyn, recodes="1=0; 2=1; else=NA")

sub$race_eth[sub$hisp == 0 & sub$race=="wht"]<-"NHWhite"
```

```
Warning: Unknown or uninitialised column: `race_eth`.
```

```
sub$race_eth[sub$hisp == 0 & sub$race=="blk"]<-"NHBlack"
sub$race_eth[sub$hisp == 0 & sub$race=="other"]<-"NHother"
sub$race_eth[sub$hisp == 1 ]<-"Hispanic"
sub$race_eth[is.na(sub$hisp) ==T | is.na(sub$race)==T]<-NA

library(forcats)

sub$race_eth<-fct_relevel(sub$race_eth, c("NHWhite", "NHBlack" , "Hispanic", "NHother"))
```

Create person - period file

```
subpp<-survSplit(Surv(d.age, d.event)~., data=sub,
cut=seq(18, 100, 5),  episode="mort_5_yr")
```

Analysis of death outcome

```
des2<-svydesign(ids=~psu, strata=~strata,
                weights = ~perweight, data=subpp, nest=T)

# constant model
m0<-svyglm(d.event~ race_eth+college,
           design=des2, family=binomial (link="cloglog"))
```

```
Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```
#general model
m1<-svyglm(d.event~factor(mort_5_yr) + race_eth+college,
           design=des2, family=binomial (link="cloglog"))
```

```
Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```
#linear model
m2<-svyglm(d.event~mort_5_yr+race_eth+college, design=des2,
           family=binomial (link="cloglog"))
```

```
Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```
#quadratic model
m3<-svyglm(d.event~mort_5_yr+ I(mort_5_yr^2)+race_eth+college,
           design=des2, family=binomial (link="cloglog"))
```

```
Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```
#spline model
library(splines)
m4<-svyglm(d.event~ns(mort_5_yr, df=3)+race_eth+college, design=des2,
           family=binomial (link="cloglog"))
```

```
Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

Generate hazard plots

```
newdat<-expand.grid(mort_5_yr = unique(subpp$mort_5_yr),
                    race_eth= unique(subpp$race_eth),
                    college = unique(subpp$college))

newdat<-newdat%>%
  dplyr::filter(complete.cases(.))

newdat$h0<-predict(m0, newdata=newdat, type="response")
newdat$h1<-predict(m1, newdata=newdat, type="response")
newdat$h2<-predict(m2, newdata=newdat, type="response")
newdat$h3<-predict(m3, newdata=newdat, type="response")
newdat$h4<-predict(m4, newdata=newdat, type="response")

head(newdat)
```

```
  mort_5_yr race_eth    college         h0           h1           h2
1         1 Hispanic hs or less 0.01454309 3.999548e-05 0.0002484581
2         2 Hispanic hs or less 0.01454309 7.439959e-04 0.0004353330
3         3 Hispanic hs or less 0.01454309 1.642436e-03 0.0007627100
4         4 Hispanic hs or less 0.01454309 2.140502e-03 0.0013361151
5         5 Hispanic hs or less 0.01454309 3.376545e-03 0.0023401008
6         6 Hispanic hs or less 0.01454309 5.311995e-03 0.0040969538
            h3           h4
1 0.0008750067 0.0001388837
2 0.0011530416 0.0002166392
3 0.0015674471 0.0003353653
4 0.0021980510 0.0005113228
5 0.0031794367 0.0007620237
6 0.0047432881 0.0011016385
```

```
library(data.table)
```

```
Attaching package: 'data.table'

The following objects are masked from 'package:dplyr':

    between, first, last

The following object is masked from 'package:purrr':

    transpose
```

```
library(magrittr)
```

```
Attaching package: 'magrittr'

The following object is masked from 'package:purrr':

    set_names

The following object is masked from 'package:tidyr':

    extract
```

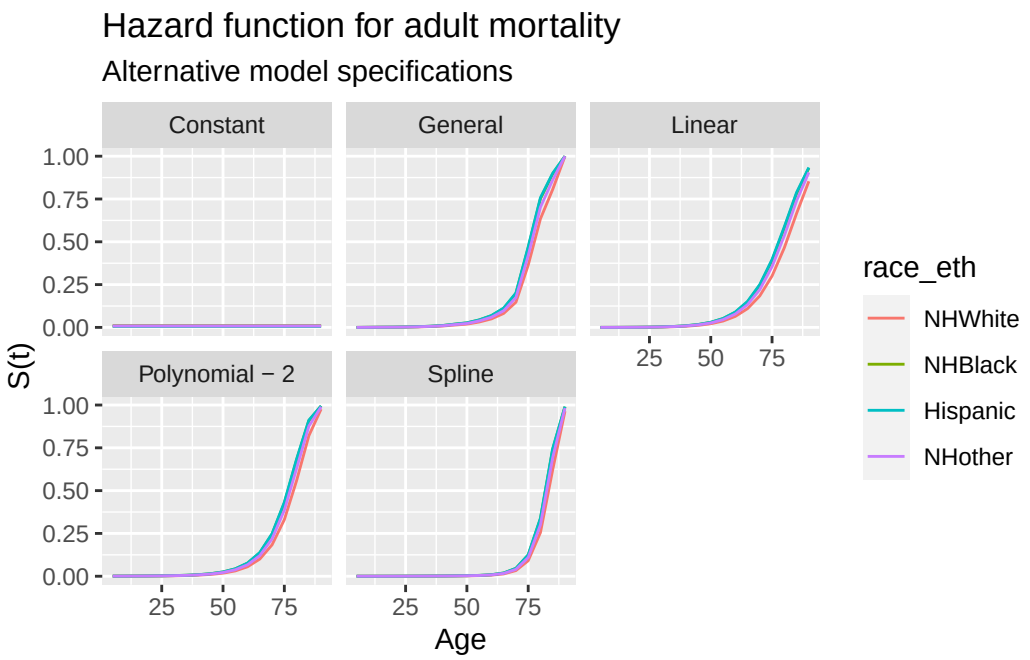
```
out<-melt(setDT(newdat),
          id = c("mort_5_yr", "race_eth", "college"),
          measure.vars = list(haz=c("h0", "h1","h2","h3", "h4")))
head(out, n=20)
```

```
    mort_5_yr race_eth    college variable      value
 1:         1 Hispanic hs or less       h0 0.01454309
 2:         2 Hispanic hs or less       h0 0.01454309
 3:         3 Hispanic hs or less       h0 0.01454309
 4:         4 Hispanic hs or less       h0 0.01454309
 5:         5 Hispanic hs or less       h0 0.01454309
 6:         6 Hispanic hs or less       h0 0.01454309
 7:         7 Hispanic hs or less       h0 0.01454309
 8:         8 Hispanic hs or less       h0 0.01454309
 9:         9 Hispanic hs or less       h0 0.01454309
10:        10 Hispanic hs or less       h0 0.01454309
11:        11 Hispanic hs or less       h0 0.01454309
12:        12 Hispanic hs or less       h0 0.01454309
13:        13 Hispanic hs or less       h0 0.01454309
14:        14 Hispanic hs or less       h0 0.01454309
15:        15 Hispanic hs or less       h0 0.01454309
16:        16 Hispanic hs or less       h0 0.01454309
17:        17 Hispanic hs or less       h0 0.01454309
18:        18 Hispanic hs or less       h0 0.01454309
19:         1  NHother hs or less       h0 0.01876243
20:         2  NHother hs or less       h0 0.01876243
```

```
library(ggplot2)

out%>%
  dplyr::filter(college == "coll")%>%
  dplyr::mutate(mod = dplyr::case_when(.$variable =="h0" ~ "Constant",
                           .$variable =="h1" ~ "General",
                           .$variable =="h2" ~ "Linear",
                           .$variable =="h3" ~ "Polynomial - 2",
                           .$variable =="h4" ~ "Spline"))%>%
  ggplot(aes(x = mort_5_yr*5, y=value ))+
  geom_line(aes(group=race_eth, color=race_eth) )+
  labs(title = "Hazard function for adult mortality",
       subtitle = "Alternative model specifications")+
  xlab("Age")+ylab("S(t)")+
```

```
  facet_wrap(~mod)#+ scale_y_continuous(trans = "log10")
```

```
Don't know how to automatically pick scale for object of type svystat. Defaulting to continuo
```



Model fits

Here we construct a table of the relative model fits for the five different time specifications

```
  AIC0<-AIC(m0)
```

```
Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```
  AIC1<-AIC(m1)
```

```
Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```
  AIC2<-AIC(m2)
```

```
Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```
AIC3<-AIC(m3)
```

```
Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```
AIC4<-AIC(m4)
```

```
Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```
AICS<-data.frame(AIC = c(AIC0["AIC"],AIC1["AIC"],AIC2["AIC"],AIC3["AIC"],AIC4["AIC"]),
                 mod = factor(c("const", "general", "linear", "poly 2", "spline")) )

AICS$mod<-forcats::fct_relevel(AICS$mod,
                               c("general", "const" , "linear", "poly2", "spline"))
```

```
Warning: 1 unknown level in `f`: poly2
```

```
AICS$deltaAIC<-AICS$AIC - AICS$AIC[AICS$mod=="general"]

knitr::kable(AICS[, c("mod", "AIC", "deltaAIC")],
             caption = "Relative AIC for alternative time specifications")
```

Table 2: Relative AIC for alternative time specifications

mod	AIC	deltaAIC
const	121377.88	44304.9492
general	77072.93	0.0000
linear	77954.86	881.9238
poly 2	77490.12	417.1851
spline	77400.06	327.1259

We see that nothing touches the AIC for the general model in this case.