# Survival Analysis (SI) Lab 5

*Albert C Liu*

*2019.06.14*

For those who haven't. Do this first.

```
#install.packages("tidyverse")
```

## Import the recid dataset

```
# set working directory (Rmarkdown will give you an error message but it's fine)
setwd("C:/Users/user/Dropbox/JHU/MHS Term 1/Survival Analysis/SI Course Materials for Albert/datasets/")
# read the data file
load("recid.rdata")
```

## Wide data format

```
# this is just for renaming
recid_wide = recid
rm(recid)
```

### Create survival (time-to-event) object: same as lab 4

```
library(survival)
survobj_wide = Surv(recid_wide$week, recid_wide$arrest) #I didn't do recid_wide$survobj
```

### Cox PH model: same as lab 4

Applying the summary() function gives more information:

- Estimates of the betas-coef, HRs-exp(coef), SEs, and p-values for each beta (test H0: beta=0)
- Estimate of the HR with confidence bounds
- p-values for likelihood ratio, Wald and score tests for the global null (H0: betai=0 for all i).

```
summary(coxph(survobj_wide ~ fin + age + race + wexp + mar + paro + prio,
              data = recid_wide))
```

```
## Call:
## coxph(formula = survobj_wide ~ fin + age + race + wexp + mar +
##     paro + prio, data = recid_wide)
##
##   n= 432, number of events= 114
##
##          coef exp(coef) se(coef)      z Pr(>|z|)
## fin  -0.37942   0.68426  0.19138 -1.983  0.04742 *
```

```
## age  -0.05744   0.94418  0.02200 -2.611  0.00903 **
## race  0.31390   1.36875  0.30799  1.019  0.30812
## wexp -0.14980   0.86088  0.21222 -0.706  0.48029
## mar  -0.43370   0.64810  0.38187 -1.136  0.25606
## paro -0.08487   0.91863  0.19576 -0.434  0.66461
## prio  0.09150   1.09581  0.02865  3.194  0.00140 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##      exp(coef) exp(-coef) lower .95 upper .95
## fin     0.6843     1.4614    0.4702    0.9957
## age     0.9442     1.0591    0.9043    0.9858
## race    1.3688     0.7306    0.7484    2.5032
## wexp    0.8609     1.1616    0.5679    1.3049
## mar     0.6481     1.5430    0.3066    1.3699
## paro    0.9186     1.0886    0.6259    1.3482
## prio    1.0958     0.9126    1.0360    1.1591
##
## Concordance= 0.64  (se = 0.027 )
## Rsquare= 0.074   (max possible= 0.956 )
## Likelihood ratio test= 33.27  on 7 df,   p=2e-05
## Wald test            = 32.11  on 7 df,   p=4e-05
## Score (logrank) test = 33.53  on 7 df,   p=2e-05
```

# Data Transformation

tidyr and dplyr are both part of the tidyverse (so is ggplot2 and readr)

https://www.tidyverse.org/packages/

https://www.rstudio.com/wp-content/uploads/2015/02/data-wrangling-cheatsheet.pdf

The tidyr package helps with transforming data frames between wide and long formats

- gather - wide to long
  - gathers multiple columns and essential sticks them into one column
  - the names of multiple columns become levels of a single variable
- spread - long to wide
  - takes a single variable (with multiple levels) and spreads them across multiple columns
  - (useful if you want to compute a statistic across multiple levels/variables)

The dplyr package is a super useful tool in data manipulation

- select: return a subset of the columns of a data frame
- filter: extract a subset of rows from a data frame based on logical conditions
- arrange: reorder rows of a data frame
- rename: rename variables in a data frame
- mutate: add new variables/columns or transform existing variables
- summarise / summarize: generate summary statistics of different variables in the data frame, possibly within strata (need to group_by first)
- group_by: stratify, paired wit mutate or summarize (generate summary statistics by stratum)

Note that tidyverse system don't like Surv object

```
# assign case id (if not already exist in the dataset)
recid_wide$id = 1:nrow(recid_wide)
```

```r
# load these 2 packages
library(tidyr); library(dplyr)
```

```
## Warning: package 'tidyr' was built under R version 3.5.3
```

```
## Warning: package 'dplyr' was built under R version 3.5.3
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
# the core is the gather() function, which takes:
## data: your (wide) dataset (has to be a "data.frame" or a "tibble")
## key: a new variable you'll create to store the variable names you gathered
## value: a new variable you'll create to store the values of the variables
## the last argument(s): the variables that you want to gather
gather(data=recid_wide, key="emp_wk", value="emp", emp1:emp52) #or 11:62
```

```
## # A tibble: 22,464 x 13
##     week arrest   fin   age  race  wexp   mar  paro  prio  educ    id
##    <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <int>
##  1    20      1     0    27     1     0     0     1     3     3     1
##  2    17      1     0    18     1     0     0     1     8     4     2
##  3    25      1     0    19     0     1     0     1    13     3     3
##  4    52      0     1    23     1     1     1     1     1     5     4
##  5    52      0     0    19     0     1     0     1     3     3     5
##  6    52      0     0    24     1     1     0     0     2     4     6
##  7    23      1     0    25     1     1     1     1     0     4     7
##  8    52      0     1    21     1     1     0     1     4     3     8
##  9    52      0     0    22     1     0     0     0     6     3     9
## 10    52      0     0    20     1     1     0     0     0     5    10
## # ... with 22,454 more rows, and 2 more variables: emp_wk <chr>, emp <dbl>
```

## This is the overall data manipulation codes:

The tidyr and dplyr functions all take in a dataset and some arguements for manipulation, and output a (manipulated) dataset. The pipeline operator %>% is used to pass on the result (a dataset) of the previous function, and pass it to the next function, which is very useful in serial data manipulation. When using %>% to connect the functions, you omit the 1st arguement–the name of the dataset in the following functions.

In the 4th line of the following chunk, sub() is a base function (so it doesn't need to import library) that can substitute a character string to another, and as.numeric() is a base function that "coerce" the data type into numeric. So that line essentially do is something like "emp1" >> "1" >> 1.

```r
recid_long <- recid_wide %>%
    gather(emp_wk, emp, emp1:emp52) %>% #transfer wide to long
    arrange(id) %>% #sort by id
    mutate(tf = as.numeric(sub("emp", "", emp_wk))) %>% #final time
    mutate(ti = tf-1) %>% #initial time (for each time interval)
```

```
    filter(tf<=week) %>% #keep only the intervals before arrest/censoring
    mutate(event = ifelse(tf<week, 0, arrest)) #censoring indicator for each time interval
```

Note that there are 432 participatns, but now you have 19809 rows (observations) in the long format dataset, each row reflects a time interval (in this case, every interval is one week) under follow-up of a participant.

These are the equivalent codes as the previous chunk:

```
recid_1 = gather(recid_wide, emp_wk, emp, emp1:emp52)
recid_2 = arrange(recid_1, id)
recid_3 = mutate(recid_2, tf = as.numeric(sub("emp", "", emp_wk)))
recid_4 = mutate(recid_3, ti = tf-1)
recid_5 = filter(recid_4, tf<=week)
recid_6 = mutate(recid_5, event = ifelse(tf<week, 0, arrest))
recid_long2 <- recid_6

# if you want to save some memory for your computer,
# you can remove the intermediate datasets using rm() function
rm(recid_1, recid_2, recid_3, recid_4, recid_5, recid_6)
```

# Long data format

## Create survival (time-to-event) object: this is the trick!

Note that this time, you need to input 3 arguments: time (start of each time interval), time2 (end of each time interval), and event (censoring indicator). Read more on:

https://www.rdocumentation.org/packages/survival/versions/2.44-1.1/topics/Surv

```
survobj_long = Surv(time=recid_long$ti, time2=recid_long$tf,
                    event=recid_long$event, type="counting")
```

## Cox PH model: basically the same as lab 4

Note that this model allows time-varying exposure for emp but assumes that the effect size of emp is the same in different interval.

```
summary(coxph(survobj_long ~ fin + age + race + wexp + mar + paro + prio + emp,
              data = recid_long))
```

```
## Call:
## coxph(formula = survobj_long ~ fin + age + race + wexp + mar +
##     paro + prio + emp, data = recid_long)
##
##   n= 19809, number of events= 114
##
##          coef exp(coef) se(coef)      z Pr(>|z|)
## fin  -0.35672   0.69997  0.19113 -1.866  0.06198 .
## age  -0.04634   0.95472  0.02174 -2.132  0.03301 *
## race  0.33866   1.40306  0.30960  1.094  0.27402
## wexp -0.02555   0.97477  0.21142 -0.121  0.90380
## mar  -0.29375   0.74546  0.38303 -0.767  0.44314
## paro -0.06421   0.93781  0.19468 -0.330  0.74156
```

```
## prio  0.08514   1.08887  0.02896  2.940  0.00328 **
## emp  -1.32832   0.26492  0.25072 -5.298 1.17e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##      exp(coef) exp(-coef) lower .95 upper .95
## fin     0.7000     1.4286    0.4813    1.0180
## age     0.9547     1.0474    0.9149    0.9963
## race    1.4031     0.7127    0.7648    2.5740
## wexp    0.9748     1.0259    0.6441    1.4753
## mar     0.7455     1.3414    0.3519    1.5793
## paro    0.9378     1.0663    0.6403    1.3735
## prio    1.0889     0.9184    1.0288    1.1525
## emp     0.2649     3.7747    0.1621    0.4330
##
## Concordance= 0.708  (se = 0.027 )
## Rsquare= 0.003   (max possible= 0.066 )
## Likelihood ratio test= 68.65  on 8 df,   p=9e-12
## Wald test            = 56.15  on 8 df,   p=3e-09
## Score (logrank) test = 64.48  on 8 df,   p=6e-11
```

## Lag time

However, arrests probably affect employment status rather than the other way.

Solution: lag emp by, say, a week (i.e. look at the employment status of the week previous to arrest)

- lag() takes a vector and returns a vector with every value replaced by its "previous" value
- lead() takes a vector and returns a vector with every value replaced by its "next" value

Note: after you do lag/lead, you'll get a NA in the very beginning/end (of each individual, in this case), which is fine.

```
recid_long %>%
    group_by(id) %>% #group by id i.e. for each participant
    mutate(empprev=lag(emp, n=1)) #lag one observation
```

```
## # A tibble: 19,809 x 17
## # Groups:   id [432]
##     week arrest   fin   age  race  wexp   mar  paro  prio  educ    id
##    <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <int>
##  1    20      1     0    27     1     0     0     1     3     3     1
##  2    20      1     0    27     1     0     0     1     3     3     1
##  3    20      1     0    27     1     0     0     1     3     3     1
##  4    20      1     0    27     1     0     0     1     3     3     1
##  5    20      1     0    27     1     0     0     1     3     3     1
##  6    20      1     0    27     1     0     0     1     3     3     1
##  7    20      1     0    27     1     0     0     1     3     3     1
##  8    20      1     0    27     1     0     0     1     3     3     1
##  9    20      1     0    27     1     0     0     1     3     3     1
## 10    20      1     0    27     1     0     0     1     3     3     1
## # ... with 19,799 more rows, and 6 more variables: emp_wk <chr>,
## #   emp <dbl>, tf <dbl>, ti <dbl>, event <dbl>, empprev <dbl>
```

```
summary(coxph(survobj_long ~ fin + age + race + wexp + mar + paro + prio + emp,
              data = recid_long))
```

```
## Call:
## coxph(formula = survobj_long ~ fin + age + race + wexp + mar +
##     paro + prio + emp, data = recid_long)
##
##   n= 19809, number of events= 114
##
##          coef exp(coef) se(coef)      z Pr(>|z|)
## fin  -0.35672   0.69997  0.19113 -1.866  0.06198 .
## age  -0.04634   0.95472  0.02174 -2.132  0.03301 *
## race  0.33866   1.40306  0.30960  1.094  0.27402
## wexp -0.02555   0.97477  0.21142 -0.121  0.90380
## mar  -0.29375   0.74546  0.38303 -0.767  0.44314
## paro -0.06421   0.93781  0.19468 -0.330  0.74156
## prio  0.08514   1.08887  0.02896  2.940  0.00328 **
## emp  -1.32832   0.26492  0.25072 -5.298 1.17e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##      exp(coef) exp(-coef) lower .95 upper .95
## fin     0.7000     1.4286    0.4813    1.0180
## age     0.9547     1.0474    0.9149    0.9963
## race    1.4031     0.7127    0.7648    2.5740
## wexp    0.9748     1.0259    0.6441    1.4753
## mar     0.7455     1.3414    0.3519    1.5793
## paro    0.9378     1.0663    0.6403    1.3735
## prio    1.0889     0.9184    1.0288    1.1525
## emp     0.2649     3.7747    0.1621    0.4330
##
## Concordance= 0.708  (se = 0.027 )
## Rsquare= 0.003   (max possible= 0.066 )
## Likelihood ratio test= 68.65  on 8 df,   p=9e-12
## Wald test            = 56.15  on 8 df,   p=3e-09
## Score (logrank) test = 64.48  on 8 df,   p=6e-11
```

## Time-dependent effect

If you every come up with this question, say, does the effect of financial aid (fin) changes over time? You can add an interaction term with time (here it means time interval) to the model.

Coding: Simply change fin into fin*tf*. *As in other regression models in R, when you put variable1*variable2 in the right side of ~ symbol, the model will include the variable1, variable2, and their interaction term (variable1:variable2).

```
summary(coxph(survobj_long ~ fin*tf + age + race + wexp + mar + paro + prio + emp, data = recid_long))
```

```
## Call:
## coxph(formula = survobj_long ~ fin * tf + age + race + wexp +
##     mar + paro + prio + emp, data = recid_long)
##
##   n= 19809, number of events= 114
##
##              coef exp(coef)  se(coef)      z Pr(>|z|)
## fin    -0.402206  0.668843  0.425647 -0.945  0.34470
## tf           NA        NA  0.000000     NA       NA
```

```
## age    -0.046323  0.954733  0.021732 -2.132  0.03304 *
## race    0.338005  1.402147  0.309643  1.092  0.27501
## wexp   -0.025652  0.974674  0.211398 -0.121  0.90342
## mar    -0.293139  0.745918  0.383032 -0.765  0.44409
## paro   -0.064671  0.937376  0.194710 -0.332  0.73978
## prio    0.084964  1.088678  0.028981  2.932  0.00337 **
## emp    -1.329271  0.264670  0.250850 -5.299 1.16e-07 ***
## fin:tf  0.001584  1.001585  0.013230  0.120  0.90472
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##        exp(coef) exp(-coef) lower .95 upper .95
## fin       0.6688     1.4951    0.2904    1.5404
## tf            NA         NA        NA        NA
## age       0.9547     1.0474    0.9149    0.9963
## race      1.4021     0.7132    0.7642    2.5725
## wexp      0.9747     1.0260    0.6440    1.4750
## mar       0.7459     1.3406    0.3521    1.5803
## paro      0.9374     1.0668    0.6400    1.3729
## prio      1.0887     0.9185    1.0286    1.1523
## emp       0.2647     3.7783    0.1619    0.4327
## fin:tf    1.0016     0.9984    0.9759    1.0279
##
## Concordance= 0.708  (se = 0.027 )
## Rsquare= 0.003   (max possible= 0.066 )
## Likelihood ratio test= 68.67  on 9 df,   p=3e-11
## Wald test            = 56.18  on 9 df,   p=7e-09
## Score (logrank) test = 64.52  on 9 df,   p=2e-10
```

The interaction term is not significant in this case, which means the HR of financial aid on the outcome (getting arrested) does not change over time => proportional hazard assumption holds for financial aid. If that term is significant, then we would say that the proportional hazard assumption does not hold for this variable.

You will also find out that the estimates of the time variable (tf, in this case) is NA, which is expected, because Cox PH model innately controls for the time metric you used, which is basically the case for survival analysis as I have alluded earlier in lab2.

- Mathematic Intuition: Remember the partial likelihood that you wrote for HW4? Every time you constructed a risk set, you "matched" the time variable. That's why the time variable cannot contribute any information to the partial MLE.

If you don't want to see the NA term, then you keep fin there, and add the pure interaction term fin:tf.

```
summary(coxph(survobj_long ~ fin + fin:tf + age + race + wexp + mar + paro + prio + emp, data = recid_l
```

```
## Call:
## coxph(formula = survobj_long ~ fin + fin:tf + age + race + wexp +
##     mar + paro + prio + emp, data = recid_long)
##
##   n= 19809, number of events= 114
##
##             coef exp(coef)  se(coef)      z Pr(>|z|)
## fin    -0.402206  0.668843  0.425647 -0.945  0.34470
## age    -0.046323  0.954733  0.021732 -2.132  0.03304 *
## race    0.338005  1.402147  0.309643  1.092  0.27501
## wexp   -0.025652  0.974674  0.211398 -0.121  0.90342
```

```
## mar    -0.293139  0.745918  0.383032 -0.765  0.44409
## paro   -0.064671  0.937376  0.194710 -0.332  0.73978
## prio    0.084964  1.088678  0.028981  2.932  0.00337 **
## emp    -1.329271  0.264670  0.250850 -5.299 1.16e-07 ***
## fin:tf  0.001584  1.001585  0.013230  0.120  0.90472
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##        exp(coef) exp(-coef) lower .95 upper .95
## fin       0.6688     1.4951    0.2904    1.5404
## age       0.9547     1.0474    0.9149    0.9963
## race      1.4021     0.7132    0.7642    2.5725
## wexp      0.9747     1.0260    0.6440    1.4750
## mar       0.7459     1.3406    0.3521    1.5803
## paro      0.9374     1.0668    0.6400    1.3729
## prio      1.0887     0.9185    1.0286    1.1523
## emp       0.2647     3.7783    0.1619    0.4327
## fin:tf    1.0016     0.9984    0.9759    1.0279
##
## Concordance= 0.708  (se = 0.027 )
## Rsquare= 0.003   (max possible= 0.066 )
## Likelihood ratio test= 68.67  on 9 df,   p=3e-11
## Wald test            = 56.18  on 9 df,   p=7e-09
## Score (logrank) test = 64.52  on 9 df,   p=2e-10
```

## Stratified Cox PH model (can be applied in both wide and long format)

e.g. If you believe that, say, different races had different baseline hazards, i.e. the hazard functions of black and white people are not proportional (you can think of as the shape of the hazard curves are not the same), then it's more reasonable to stratify by it (otherwise, just use race as a covariate).

In R, the code is super simple, you just add a strata() function to race.

```
summary(coxph(survobj_long ~ fin + age + strata(race) + wexp + mar + paro + prio + emp,
              data = recid_long))
```

```
## Call:
## coxph(formula = survobj_long ~ fin + age + strata(race) + wexp +
##     mar + paro + prio + emp, data = recid_long)
##
##   n= 19809, number of events= 114
##
##          coef exp(coef) se(coef)      z Pr(>|z|)
## fin  -0.35027   0.70450  0.19123 -1.832  0.06700 .
## age  -0.04687   0.95421  0.02180 -2.150  0.03155 *
## wexp -0.01720   0.98294  0.21204 -0.081  0.93534
## mar  -0.31768   0.72784  0.38357 -0.828  0.40755
## paro -0.06464   0.93740  0.19484 -0.332  0.74006
## prio  0.08615   1.08997  0.02904  2.967  0.00301 **
## emp  -1.32391   0.26609  0.25063 -5.282 1.28e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
##      exp(coef) exp(-coef) lower .95 upper .95
## fin     0.7045     1.4195    0.4843    1.0248
## age     0.9542     1.0480    0.9143    0.9959
## wexp    0.9829     1.0174    0.6487    1.4894
## mar     0.7278     1.3739    0.3432    1.5436
## paro    0.9374     1.0668    0.6399    1.3733
## prio    1.0900     0.9175    1.0297    1.1538
## emp     0.2661     3.7581    0.1628    0.4349
##
## Concordance= 0.701  (se = 0.03 )
## Rsquare= 0.003   (max possible= 0.062 )
## Likelihood ratio test= 67.88  on 7 df,   p=4e-12
## Wald test            = 55.42  on 7 df,   p=1e-09
## Score (logrank) test = 63.77  on 7 df,   p=3e-11
```

Note: The partial likelihood for the stratified data is the product of the partial likelihood for each stratum.

# Estimate survival function based on fitted proportional hazards model

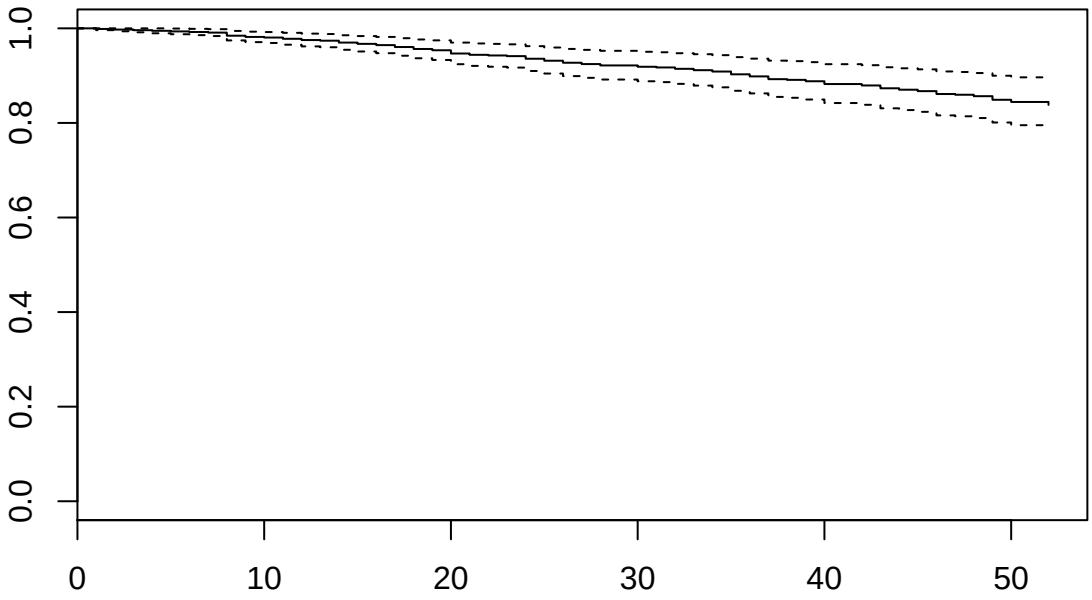
## Time-fixed effect (can be either data type)

```
# assume this is our Cox PH model, here we only include fin (received financial
# aid after release or not) and prio (number of prior convictions) as covatiates
# , but you can definitely include other/more covariates
PH1 = coxph(survobj_wide ~ fin + prio, data = recid_wide)

# create a data frame with desired covariate values (here we set to fin=1 and
# prio=0, namely received financial aid and no prior convictions, you can set
# different values)
covs1 = data.frame(fin = 1, prio = 0)

# list out the estimated survival function
summary(survfit(PH1, newdata=covs1))
```

```
## Call: survfit(formula = PH1, newdata = covs1)
##
##  time n.risk n.event survival std.err lower 95% CI upper 95% CI
##     1    432       1    0.999 0.00130        0.996        1.000
##     2    431       1    0.997 0.00187        0.994        1.000
##     3    430       1    0.996 0.00231        0.992        1.000
##     4    429       1    0.995 0.00270        0.990        1.000
##     5    428       1    0.994 0.00306        0.988        1.000
##     6    427       1    0.992 0.00339        0.986        0.999
##     7    426       1    0.991 0.00371        0.984        0.998
##     8    425       5    0.984 0.00512        0.974        0.995
##     9    420       2    0.982 0.00564        0.971        0.993
##    10    418       1    0.981 0.00590        0.969        0.992
##    11    417       2    0.978 0.00640        0.965        0.991
##    12    415       2    0.975 0.00689        0.962        0.989
##    13    413       1    0.974 0.00713        0.960        0.988
```

```
##     14    412       3    0.970 0.00784        0.955        0.985
##     15    409       2    0.967 0.00831        0.951        0.984
##     16    407       2    0.965 0.00877        0.947        0.982
##     17    405       3    0.960 0.00945        0.942        0.979
##     18    402       3    0.956 0.01013        0.937        0.976
##     19    399       2    0.954 0.01057        0.933        0.975
##     20    397       5    0.947 0.01167        0.924        0.970
##     21    392       2    0.944 0.01210        0.921        0.968
##     22    390       1    0.943 0.01232        0.919        0.967
##     23    389       1    0.941 0.01254        0.917        0.966
##     24    388       4    0.936 0.01340        0.910        0.962
##     25    384       3    0.932 0.01403        0.904        0.959
##     26    381       3    0.927 0.01466        0.899        0.956
##     27    378       2    0.924 0.01508        0.895        0.954
##     28    376       2    0.922 0.01550        0.892        0.952
##     30    374       2    0.919 0.01591        0.888        0.950
##     31    372       1    0.917 0.01611        0.886        0.949
##     32    371       2    0.914 0.01652        0.883        0.947
##     33    369       2    0.912 0.01693        0.879        0.945
##     34    367       2    0.909 0.01734        0.875        0.943
##     35    365       4    0.903 0.01815        0.868        0.939
##     36    361       3    0.898 0.01875        0.862        0.936
##     37    358       4    0.893 0.01956        0.855        0.932
##     38    354       1    0.891 0.01976        0.853        0.931
##     39    353       2    0.888 0.02016        0.849        0.929
##     40    351       4    0.882 0.02095        0.842        0.924
##     42    347       2    0.879 0.02135        0.838        0.922
##     43    345       4    0.873 0.02213        0.831        0.918
##     44    341       2    0.870 0.02252        0.827        0.916
##     45    339       2    0.867 0.02291        0.823        0.913
##     46    337       4    0.861 0.02369        0.816        0.909
##     47    333       1    0.860 0.02388        0.814        0.908
##     48    332       2    0.857 0.02427        0.810        0.905
##     49    330       5    0.849 0.02523        0.801        0.900
##     50    325       3    0.844 0.02580        0.795        0.896
##     52    322       4    0.838 0.02656        0.788        0.892
```

```
# you can also plot it
plot(survfit(PH1, newdata=covs1))
```




This also works if you modelled a time-varying covariate with a fixed effect (HR) over time, and want to estimate the survival function assuming that the covariate is a fixed value over time.

### Time-dependent effect (has to be counting process data)

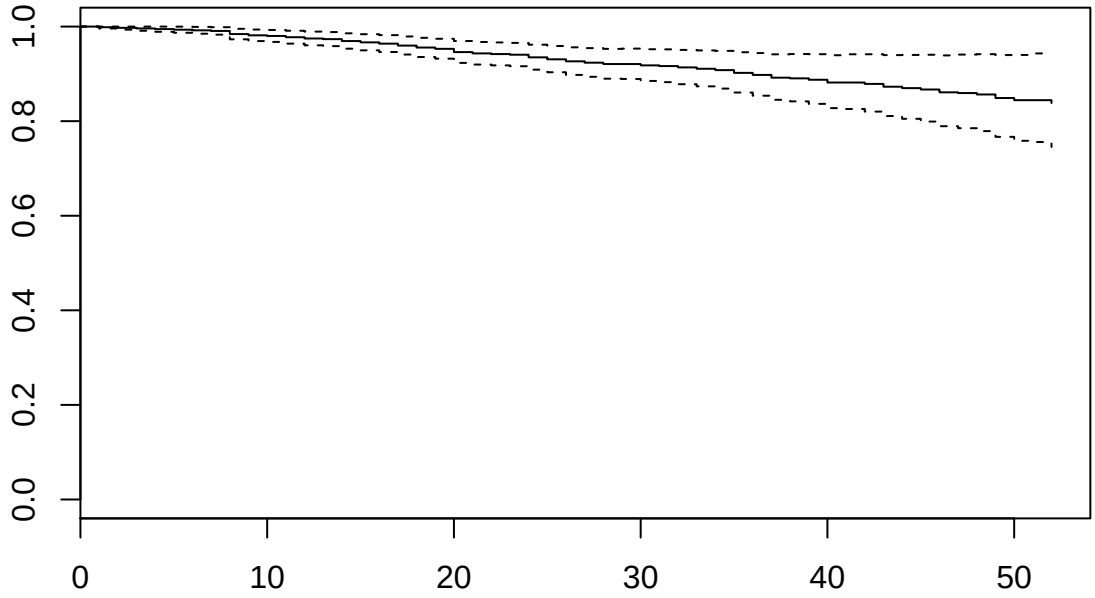
```
# assume fin has different effects in different time intervals, because later on
# we have to use survfit to estimate S(t), it is advisable to create interaction
# term ourselves before fitting the model, because you might assume different
# functional form of interaction
recid_long$fin_t = recid_long$fin * recid_long$tf

# assume this is our Cox PH model, here we only include fin (received financial
# aid after release or not) and prio (number of prior convictions) as covatiates
# , but you can definitely include other/more covariates
# it is ok to use fin*tf (you'll receive a warning but it's ok) but NOT fin:tf
PH2 = coxph(Surv(ti, tf, event) ~ fin + fin_t + prio, data = recid_long)

# take the time interval values corresponding to an individual with the last
# event time. This contains all the necessary time intervals.
last = recid_long$id[which.max(recid_long$tf)]
intervals = recid_long[recid_long$id==last, c("ti", "tf", "event")]

# create a data frame with (constant) desired covariate values (here we set to
# fin=1 and prio=0, namely received financial aid and no prior convictions, you
# can set different values)
```

```r
covs2 = data.frame(fin = 1, prio = 0, intervals)
covs2$fin_t = covs2$fin * covs2$tf

# list out the estimated survival function
summary(survfit(PH2, newdata=covs2, individual=T))
```

```
## Call: survfit(formula = PH2, newdata = covs2, individual = T)
##
##  time n.risk n.event survival std.err lower 95% CI upper 95% CI
##     1    432       1    0.999 0.00136        0.996        1.000
##     2    431       1    0.997 0.00198        0.993        1.000
##     3    430       1    0.996 0.00248        0.991        1.000
##     4    429       1    0.995 0.00292        0.989        1.000
##     5    428       1    0.993 0.00332        0.987        1.000
##     6    427       1    0.992 0.00368        0.985        0.999
##     7    426       1    0.991 0.00401        0.983        0.999
##     8    425       5    0.984 0.00568        0.973        0.995
##     9    420       2    0.981 0.00623        0.969        0.994
##    10    418       1    0.980 0.00644        0.968        0.993
##    11    417       2    0.977 0.00694        0.964        0.991
##    12    415       2    0.975 0.00742        0.960        0.989
##    13    413       1    0.973 0.00759        0.959        0.988
##    14    412       3    0.969 0.00829        0.953        0.986
##    15    409       2    0.967 0.00870        0.950        0.984
##    16    407       2    0.964 0.00910        0.946        0.982
##    17    405       3    0.960 0.00974        0.941        0.979
##    18    402       3    0.956 0.01036        0.936        0.976
##    19    399       2    0.953 0.01073        0.932        0.974
##    20    397       5    0.946 0.01179        0.923        0.969
##    21    392       2    0.943 0.01217        0.920        0.967
##    22    390       1    0.942 0.01236        0.918        0.966
##    23    389       1    0.941 0.01258        0.916        0.966
##    24    388       4    0.935 0.01346        0.909        0.962
##    25    384       3    0.931 0.01416        0.903        0.959
##    26    381       3    0.927 0.01489        0.898        0.956
##    27    378       2    0.924 0.01546        0.894        0.954
##    28    376       2    0.921 0.01608        0.890        0.953
##    30    374       2    0.918 0.01709        0.885        0.952
##    31    372       1    0.917 0.01768        0.883        0.952
##    32    371       2    0.914 0.01851        0.878        0.951
##    33    369       2    0.911 0.01939        0.874        0.950
##    34    367       2    0.908 0.02032        0.869        0.949
##    35    365       4    0.902 0.02171        0.861        0.946
##    36    361       3    0.898 0.02298        0.854        0.944
##    37    358       4    0.892 0.02451        0.845        0.941
##    38    354       1    0.890 0.02551        0.842        0.942
##    39    353       2    0.888 0.02677        0.837        0.942
##    40    351       4    0.882 0.02851        0.828        0.939
##    42    347       2    0.879 0.03091        0.820        0.941
##    43    345       4    0.873 0.03284        0.811        0.940
##    44    341       2    0.870 0.03440        0.805        0.940
##    45    339       2    0.867 0.03601        0.799        0.940
##    46    337       4    0.861 0.03815        0.789        0.939
##    47    333       1    0.859 0.03966        0.785        0.941
```

```
##     48     332       2    0.856 0.04145        0.779        0.942
##     49     330       5    0.849 0.04403        0.767        0.940
##     50     325       3    0.844 0.04622        0.759        0.940
##     52     322       4    0.838 0.05023        0.745        0.943
```

```
# you can also plot it
plot(survfit(PH2, newdata=covs2, individual=T))
```




For more information, please refer to:

https://www.jstatsoft.org/article/view/v061c01

(R: pg 9-10, SAS: pg 10-12)

# Alternative ways of building time-dependent/counting process dataset

## survSplit

survSplit() is a function from survival package, explanation and sample codes in:

https://www.rdocumentation.org/packages/survival/versions/2.44-1.1/topics/survSplit

https://www.jstatsoft.org/article/view/v061c01 (pg 7)

(This one works similar to the stsplit in Stata)

### tmerge

tmerge() is a function from survival package, explanation and sample codes in:

https://www.rdocumentation.org/packages/survival/versions/2.44-1.1/topics/tmerge

https://cran.r-project.org/web/packages/survival/vignettes/timedep.pdf

(Honestly I don't really like it myself. It's useful if you're familiar with it, but too abstract to explain.)