Recurrent events with R (Part II)

Juan R Gonzalez

10 abril 2021

Contents

1 Introduction

Objectives

- Understand the concept of survival analysis with recurrent event data
- Learn how to perform survival analysis with recurrent event data extending the Cox proportional hazard models
- Peform data analyses where the scientific question is to determine factors associated with time until re-ocurrences of a repeated event considering different covariates and taking into account event dependence and heterogeneity accross individuals

2 Multiple events per subject

In some situations in which the event of interest is not death, a patient may experience the event of interest several times over follow-up. Examples of recurrent event data could be: multiple heart attacks of coronary patients, recurrent infections in AIDS patients, multiple episodes of relapses from remission or when a patient is repeatedly readmitted in a hospital. The main feature of this kind of data is that the events are naturally ordered and are observed in a particular sequence over time, so we observe the event of interest at different times during the follow-up period. But, the last event could be not observed before the end of follow-up and in this way, we have the common feature of survival data: *censored data*.

This figure shows a graphical example of recurrent event data. Let us suppose that it is a patient with Chronic Obstructive Pulmonary Disease (COPD) data set described in the previous lecture ....

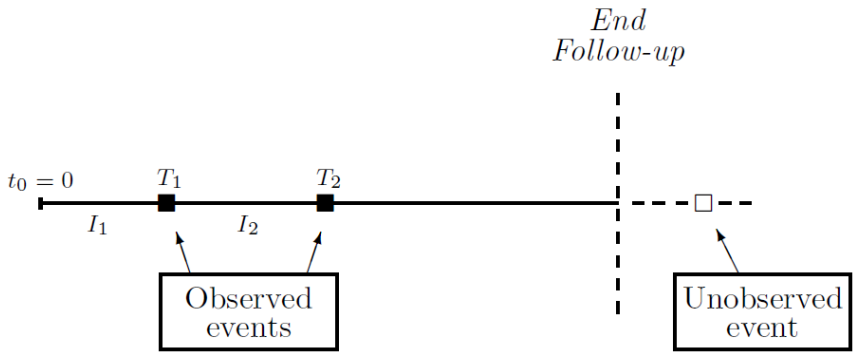


Figure 1: Graphical representation of the recurrent event process for an individual

3 Time scales

The repeated nature of a given event may produce that a subject may experience, for instance, as illustrated in the COPD example, a hospital admission several times over follow-up period. When outcome events may occur more than once over the follow-up period for a given subject, we refer as *recurrent event data*. The objective for such data is to assess the relationship of relevant predictors to the rate in which events are occurring, allowing for multiple events per

subject. Many extensions of survival models based on the Cox proportional hazards approach have been proposed to handle multiple event data. A major issue to extending proportional hazards regression models to this situation of multiple events is *intrasubject correlation*.

We consider counting process methods for analyzing time-to-event data with multiple or recurrent outcomes, using the models developed by Andersen and Gill and Prentice, Williams and Peterson, called marginal models. They are approaches proposed for adapting the Cox model to this setting that work well with our data. But we also describe alternative approach that use a *frailty model*. This last model includes a random per-subject effect that describes excess risk or *frailty* that will be further explain in another lecture.

One important aspect of multiple event data models is the time scale. Three formulations are available: gap time, total time and counting process formulation. Next figure illustrates how recurrent event data is obtained in a hypothetical study. This example will be used to describe how these three time scales are obtained. Noteice that time scale is important in multiple events analysis because defines when a subject is at risk of having an event.

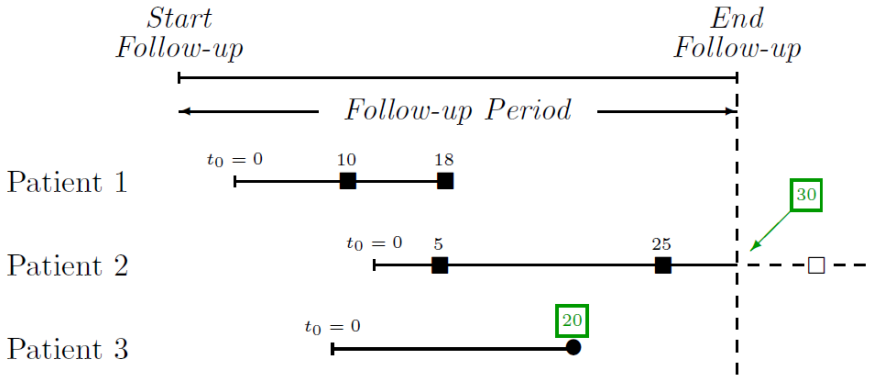


Figure 2: **Graphical representation of how data is obtained in a hypothetical study.**

This example depicts how three patiens are recorded in the data file. When follow-up period starts for a patient, it is indicated by $t_0 = 0$; $\{\blacksquare\}$ encodes the occurrence of an event and $\{\bullet\}$ correspond to censoring. Patient–1 has two events, ending the period of observation with an event; patient–2 has one event before being censored; patient–3 has no events before being censored. The first step in fitting the models is to build an adequate database. The key to construct the dataset depends on the time scale selected. Let us consider the three patients in our hypothetical example to describe the three different time scales.

3.1 Counting process

Counting process is represented as a series of observations (rows of data) with time intervals of (entry time, first event], (first event, second event],...,(kth event, last follow-up]. This is represented in the next figure

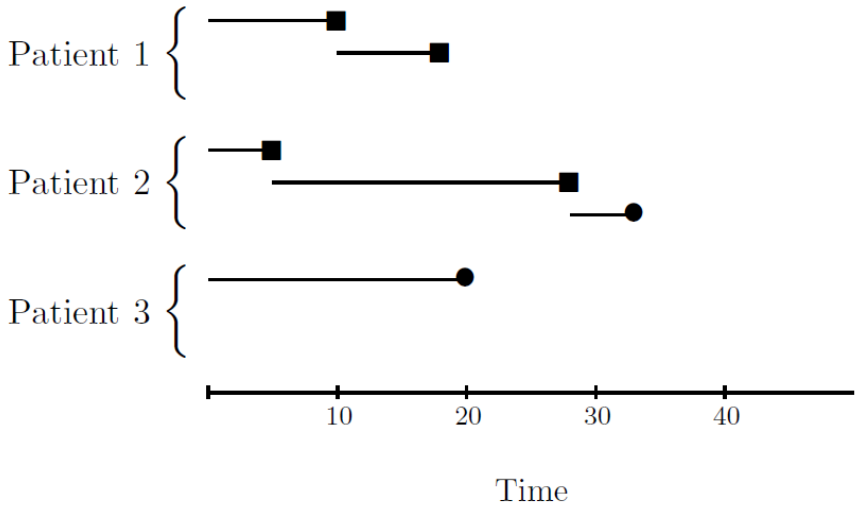


Figure 3: Graphical representation of counting process time scale of our hypothetical data example

3.2 Gap time

Gap time is the time from the last event or entry, with intervals of $(0, t_1], (0, t_2 - t_1], \cdots$. The model assumption in this case is that the gap times form a renewal process. This is represented, for our hypothetical study, in the next figure

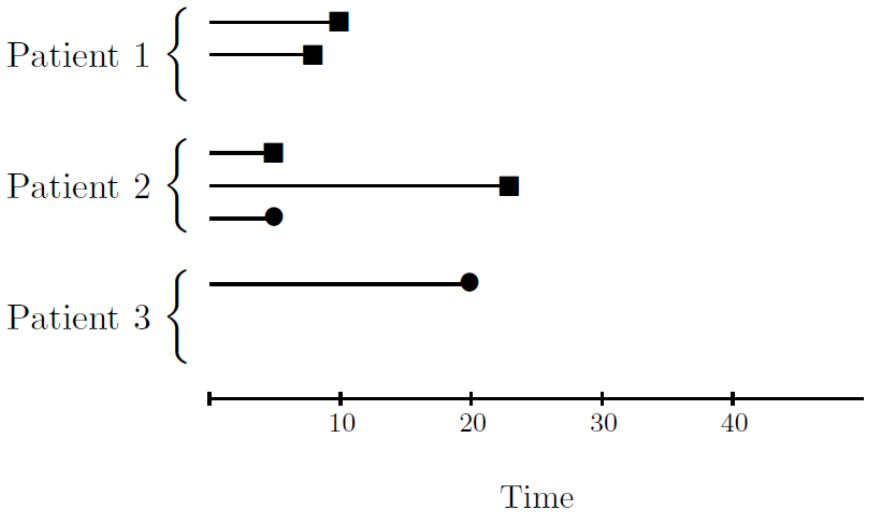


Figure 4: Graphical representation of gap time scale of our hypothetical data example

3.3 Total time

Total time corresponds to the time from the start of follow-up. This is represented, for our hypothetical study, in the next figure

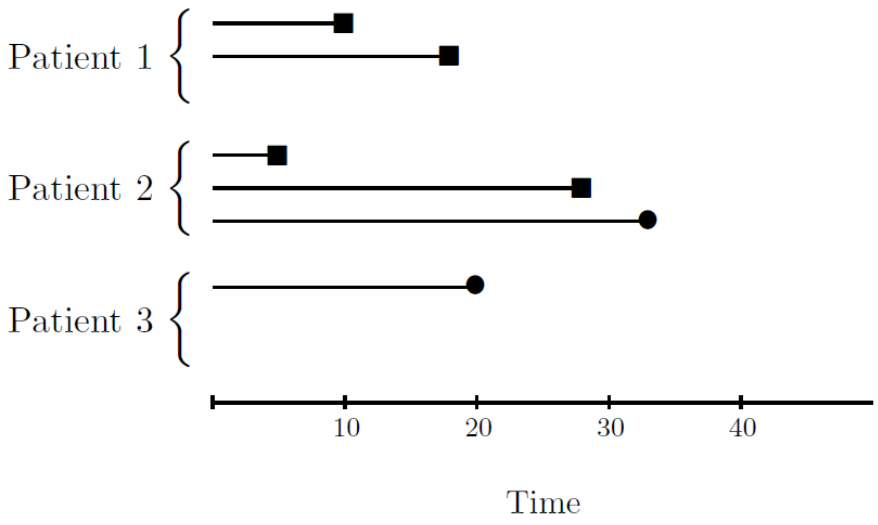


Figure 5: Graphical representation of total time scale of our hypothetical data example

A main feature for recurrent event data is that events are naturally ordered and are observed in a particular sequence over time as we have shown in the example above. However we can distinguish between multiple events of the same type (multiple hospital admissions) or multiple events of different type (hospital admission and death). The second situation will be address in another lecture.

4 Extensions of the Cox model.

The modeling of the marginal distribution for the *k*th hospital admission is based on the Cox proportional hazards model. Under this model the hazard for each *k*th event of the *i*th subject takes one of the following two forms:

$$\lambda_k(t; Z_{ik}) = \lambda_0(t) e^{\beta' Z_{ik}(t)} \quad \boxed{1}$$

$$\lambda_k(t; Z_{ik}) = \lambda_{0k}(t) e^{\beta' Z_{ik}(t)} \quad \boxed{2}$$

depending on whether the baseline hazards are ($\lambda_{0k}(t)$) or not ($\lambda_0(t)$) different for the recurrences. Z_{ik} is a covariate vector of p-dimension for the ith subject at the kth failure.

Andersen and Gill (referred to as AG model) propose to model the risk for the *k*th recurrence under the common underlying hazard given in (1) among those patients who are under observation at the time *t*. Each subject is treated as a multi-event counting process with independent increments (times between successive recurrences), so the risk of hospital admission is unaffected by any earlier admission to hospital unless explicitly expressed in the covariates, for instance the number of prior hospital admissions as a time–dependent covariate which captures the dependence structure among the recurrence times.

Prentice, Williams and Peterson (referred to as PWP model) propose to model the risk for the *k*th recurrence under the uncommon underlying hazard given in (2) among those patients who have experienced *k-1* recurrences at the time *t*. The hazard functions vary for each *k*th recurrence, $k = 1, \ldots, K$ and the dependence between recurrence times is handled through stratifying by the number of event.

The first step in fitting the models is to construct an adequate data base. The procedure is as follow. Let us consider three patients of figure 2. This information is collected in the following way in the database. Each event will produce a different row in the data base. If there is an additional follow up period after the last observed hospital admission, a new row will be added. To identify rows corresponding to recurrent events , we define the variable *status*, that will take the value 1 if the event is observed and value 0 if the time interval is censored. Moreover, we define the variable *event* to enumerate the rows of each event belonging to the same patient. In this way we obtain table 1. Note that risk intervals are defined using the counting process time scale by means of the variables *time-1* and *time-2*. Notice that counting process form may be used for *time-dependent* covariates.

Let us illustrate how to fit marginal models by using colorectal cancer hospital readmissions dataset described in the previous lecture. Data is loaded by:

Patient	*Time-1*	*Time-2*	status	*event*
1	0	10	1	1
1	10	18	1	2
2	0	5	1	1
2	5	25	1	2
2	25	30	0	3
3	0	14	0	1

Table 1: Data base to perform models AG and PWP.

```
data(colon, package="survrec")
head(colon)
##      hc time event chemoter dukes distance
## 1  5634   24     1        2     3        1
## 2  5634  433     1        2     3        1
## 3  5634  580     0        2     3        1
## 4 10767  489     1        1     2        1
## 5 10767  693     0        1     2        1
## 6 15843   15     1        1     2        1
```

Here, we have data times in total time formulation (e.g. calendar time - when patient is re-hospitalized from the begining of his/her follow-up) which is the usual way of encoding recurrent event data. We need to create two variables containing the same information as counting process. This can be done by using this function:

```
getCountingProcess <- function(x, colID, colTime) {
     id <- x[, colID]
     ids <- unique(id)
     tt <- x[,colTime]
     out <- NULL
     for (i in 1:length(ids))
       {
         tt.i <- tt[id%in%ids[i]]
         start <- c(0, tt.i[-length(tt.i)])
         out.i <- cbind(start, tt.i)
         out <- rbind(out, out.i)
         }
     ans <- data.frame(x[,colID,drop=TRUE], x[,colTime], start=out[,1],
                       stop=out[,2], x[,-c(colID, colTime)])
     names(ans)[1:2] <- names(x[,c(colID, colTime)])
     ans
     }
```

In our example, we create a new dataset having two new variables encodind counting process data as:

```
colon2 <- getCountingProcess(colon, colID=1, colTime=2)
head(colon2)
##      hc time start stop event chemoter dukes distance
## 1  5634   24     0   24     1        2     3        1
## 2  5634  433    24  433     1        2     3        1
## 3  5634  580   433  580     0        2     3        1
## 4 10767  489     0  489     1        1     2        1
## 5 10767  693   489  693     0        1     2        1
## 6 15843   15     0   15     1        1     2        1
```

Here we can observe as firt individual (hc=5634) is having two re-hospitalizations at time 24 and 433 and then is followed-up until time day 580 without observing another hospital admission. Therefore, the counting process scale is (0, 24], (24, 433] and (433, 580] and the event variable is 1, 1, 0, respectively. AG model to evaluate whether Dukes stage is associated with hospital readmission time is fitted by extending the Cox model implemented in the `survival` package:

```
library(survival)
ag.fit <- coxph(Surv(start, stop, event) ~ as.factor(dukes)+
                  cluster(hc), data=colon2)
ag.fit
## Call:
## coxph(formula = Surv(start, stop, event) ~ as.factor(dukes),
##     data = colon2, cluster = hc)
##
##                      coef exp(coef) se(coef) robust se     z        p
## as.factor(dukes)2 0.4584    1.5815   0.1323    0.1499 3.057  0.00224
## as.factor(dukes)3 1.2935    3.6456   0.1545    0.1767 7.319 2.49e-13
##
## Likelihood ratio test=64.43  on 2 df, p=1.021e-14
## n= 598, number of events= 313
##    (263 observations deleted due to missingness)
```

The AG model provides efficient inference for a covariate effect than simple Cox model of the time to the first event but requieres much stronger assumption. The AG model assumes that the data are a set of independent increments. However, when a subject may contribute multiple events this assumption could not hold. An appropriate correction is available estimating a *robust variance* and replacing the standard variance estimate with that which is corrected for the possible correlations. The `cluster` term in the model is the way to obtain the column `robust se` of the robust standard errors. The variable `id` contains the subject identifier. The AG model indicates that Duke stages 2 and 3 are

significant covariates. In choosing a model for the time to recurrent admission to hospital is important consider the process of the disease. If after undergoing the first hospital admission, the risk of the next admission to hospital may increase, this suggest a model that incorporates a time-dependent covariate for the number of hospital admission or use a model containing separate strata for each event like PWP model. However if the risk of hospital admission remains constant regardless of the number of previous admissions, the AG model would be suitable.

In order to investigate event dependence we have to create a time–dependent variable (*enum*) that encodes the number of hospital admissions. This variable can be created by using this function:

```
getEnum <- function(x, colID) {
  id <- x[, colID]
  ids <- unique(id)
  out <- NULL
  for (i in 1:length(ids))
  {
    tt.i <- sum(id%in%ids[i])
    out.i <- 1:tt.i
    out <- c(out, out.i)
  }
  out
}
```

```
colon2$enum <- getEnum(colon2, colID=1)
head(colon2)
##      hc time start stop event chemoter dukes distance enum
## 1  5634   24     0   24     1        2     3        1    1
## 2  5634  433    24  433     1        2     3        1    2
## 3  5634  580   433  580     0        2     3        1    3
## 4 10767  489     0  489     1        1     2        1    1
## 5 10767  693   489  693     0        1     2        1    2
## 6 15843   15     0   15     1        1     2        1    1
```

Then, let us incorporate this time-dependent variable into the model of re-hospitalizations in order to analyze the dependence among the recurrence times. This analysis is normally performed by considering a maximum number of recurrences that assures enough sample size. In that case we are analyzing up to the 5th hospitalization.

```
ag.fit.5 <- coxph(Surv(start, stop, event) ~ as.factor(dukes)+
                  cluster(hc), data=colon2)
ag.fit.5
```

```
## Call:
## coxph(formula = Surv(start, stop, event) ~ as.factor(dukes),
##     data = colon2, cluster = hc)
##
##                     coef exp(coef) se(coef) robust se     z        p
## as.factor(dukes)2 0.4584    1.5815   0.1323    0.1499 3.057  0.00224
## as.factor(dukes)3 1.2935    3.6456   0.1545    0.1767 7.319 2.49e-13
##
## Likelihood ratio test=64.43  on 2 df, p=1.021e-14
## n= 598, number of events= 313
##    (263 observations deleted due to missingness)
```

```
ag.fit.5.dep <- coxph(Surv(start, stop, event) ~ as.factor(dukes)+
                  cluster(hc) + enum, data=colon2)
ag.fit.5.dep
## Call:
## coxph(formula = Surv(start, stop, event) ~ as.factor(dukes) +
##     enum, data = colon2, cluster = hc)
##
##                      coef exp(coef) se(coef) robust se     z        p
## as.factor(dukes)2 0.43664   1.54749  0.13238   0.14285 3.057  0.00224
## as.factor(dukes)3 1.15327   3.16855  0.16074   0.17085 6.750 1.48e-11
## enum              0.10614   1.11198  0.01844   0.02316 4.582 4.60e-06
##
## Likelihood ratio test=86.18  on 3 df, p=< 2.2e-16
## n= 598, number of events= 313
##    (263 observations deleted due to missingness)
```

Including the time-dependent variable that accounts for the number of previous readmissions (*enum*) reduces the value of the statistic $-2\log\hat{L}$ (e.g. the Deviance) that measures the goodness-of-fit of a model

```
deviance.1 <- -2*summary(ag.fit.5)$loglik[2]
deviance.2 <- -2*summary(ag.fit.5.dep)$loglik[2]
deviance.2 - deviance.1
## [1] -21.75296
```

the reduction is statistically significant at the 1% level.

```
pchisq(deviance.1 - deviance.2, df=1, lower=FALSE)
## [1] 3.101111e-06
```

So the risk of hospital admissions increases 12% (exp(0.1061)=1.11) as the number of prior hospital admissions increases. The AG model assumes a common baseline risk function for all admissions. So, we may think that using a stratified model (PWP model) per hospital admission number could be more appropriate. This is estimating the **PWG total time model** since the `stop` variables is annotated in the calendar time.

```
hc    time start stop event
5634    24     0   24     1
5634   433    24  433     1
5634   580   433  580     0
10767  489     0  489     1
10767  693   489  693     0
...
```

The **PWG gap time model** would require to have data as

```
hc    time start stop event
5634    24     0   24     1
5634   433     0  409     1
5634   580     0  147     0
10767  489     0  489     1
10767  693     0  204     0
...
```

This is PWP total time

```
pwp.fit <- coxph(Surv(start, stop, event) ~ as.factor(dukes)+
                 cluster(hc) + strata(enum), data=colon2)
pwp.fit
## Call:
## coxph(formula = Surv(start, stop, event) ~ as.factor(dukes) +
##     strata(enum), data = colon2, cluster = hc)
##
##                      coef exp(coef) se(coef) robust se     z        p
## as.factor(dukes)2 0.3769    1.4578   0.1357    0.1378 2.735  0.00624
## as.factor(dukes)3 1.1598    3.1894   0.1652    0.1568 7.399 1.37e-13
##
## Likelihood ratio test=45.21  on 2 df, p=1.523e-10
## n= 598, number of events= 313
##    (263 observations deleted due to missingness)
```

If the primary interest is the overall importance of the covariates, regardless of the number of events or if it can be assumed that the importance of covariates is independent of the number of events, these two last models are appropriated. Both models (AG model with time-dependent variable or PWP model) give

similar results. However, if it is important the coefficient of covariates for patients in different strata or patients who had different numbers of recurrent events (hospital admissions), we need regression coefficients per-stratum, it is possible using PWP model as follow:

```
pwp.fit.strata <- coxph(Surv(start, stop, event) ~ as.factor(dukes)* strata(enum) +
                  cluster(hc) , data=colon2)
pwp.fit.strata
## Call:
## coxph(formula = Surv(start, stop, event) ~ as.factor(dukes) +
##     strata(enum) + as.factor(dukes):strata(enum), data = colon2,
##     cluster = hc)
##
##                                          coef  exp(coef)   se(coef)
## as.factor(dukes)2                    3.986e-01  1.490e+00  1.598e-01
## as.factor(dukes)3                    1.340e+00  3.818e+00  1.954e-01
## as.factor(dukes)2:strata(enum)enum=2  2.244e-02  1.023e+00  4.056e-01
## as.factor(dukes)3:strata(enum)enum=2 -4.598e-01  6.314e-01  5.072e-01
## as.factor(dukes)2:strata(enum)enum=3 -4.411e-01  6.433e-01  5.085e-01
## as.factor(dukes)3:strata(enum)enum=3 -4.034e-01  6.680e-01  5.664e-01
## as.factor(dukes)2:strata(enum)enum=4  1.452e-01  1.156e+00  7.375e-01
## as.factor(dukes)3:strata(enum)enum=4 -8.989e-01  4.070e-01  7.645e-01
## as.factor(dukes)2:strata(enum)enum=5  5.322e-01  1.703e+00  1.138e+00
## as.factor(dukes)3:strata(enum)enum=5 -3.237e-01  7.234e-01  1.181e+00
## as.factor(dukes)2:strata(enum)enum=6 -1.549e+01  1.878e-07  1.337e+03
## as.factor(dukes)3:strata(enum)enum=6  1.415e+01  1.404e+06  2.315e+03
## as.factor(dukes)2:strata(enum)enum=7         NA         NA  0.000e+00
## as.factor(dukes)3:strata(enum)enum=7         NA         NA  0.000e+00
## as.factor(dukes)2:strata(enum)enum=8         NA         NA  0.000e+00
## as.factor(dukes)3:strata(enum)enum=8         NA         NA  0.000e+00
## as.factor(dukes)2:strata(enum)enum=9         NA         NA  0.000e+00
## as.factor(dukes)3:strata(enum)enum=9         NA         NA  0.000e+00
## as.factor(dukes)2:strata(enum)enum=10        NA         NA  0.000e+00
## as.factor(dukes)3:strata(enum)enum=10        NA         NA  0.000e+00
## as.factor(dukes)2:strata(enum)enum=11        NA         NA  0.000e+00
## as.factor(dukes)3:strata(enum)enum=11        NA         NA  0.000e+00
## as.factor(dukes)2:strata(enum)enum=12        NA         NA  0.000e+00
## as.factor(dukes)3:strata(enum)enum=12        NA         NA  0.000e+00
## as.factor(dukes)2:strata(enum)enum=13        NA         NA  0.000e+00
## as.factor(dukes)3:strata(enum)enum=13        NA         NA  0.000e+00
## as.factor(dukes)2:strata(enum)enum=14        NA         NA  0.000e+00
## as.factor(dukes)3:strata(enum)enum=14        NA         NA  0.000e+00
## as.factor(dukes)2:strata(enum)enum=15        NA         NA  0.000e+00
## as.factor(dukes)3:strata(enum)enum=15        NA         NA  0.000e+00
```

```
## as.factor(dukes)2:strata(enum)enum=16         NA         NA  0.000e+00
## as.factor(dukes)3:strata(enum)enum=16         NA         NA  0.000e+00
## as.factor(dukes)2:strata(enum)enum=17         NA         NA  0.000e+00
## as.factor(dukes)3:strata(enum)enum=17         NA         NA  0.000e+00
## as.factor(dukes)2:strata(enum)enum=18         NA         NA  0.000e+00
## as.factor(dukes)3:strata(enum)enum=18         NA         NA  0.000e+00
## as.factor(dukes)2:strata(enum)enum=19         NA         NA  0.000e+00
## as.factor(dukes)3:strata(enum)enum=19         NA         NA  0.000e+00
## as.factor(dukes)2:strata(enum)enum=20         NA         NA  0.000e+00
## as.factor(dukes)3:strata(enum)enum=20         NA         NA  0.000e+00
## as.factor(dukes)2:strata(enum)enum=21         NA         NA  0.000e+00
## as.factor(dukes)3:strata(enum)enum=21         NA         NA  0.000e+00
## as.factor(dukes)2:strata(enum)enum=22         NA         NA  0.000e+00
## as.factor(dukes)3:strata(enum)enum=22         NA         NA  0.000e+00
## as.factor(dukes)2:strata(enum)enum=23         NA         NA  0.000e+00
## as.factor(dukes)3:strata(enum)enum=23         NA         NA  0.000e+00
##                                       robust se       z        p
## as.factor(dukes)2                     1.613e-01   2.471   0.0135
## as.factor(dukes)3                     1.798e-01   7.450 9.36e-14
## as.factor(dukes)2:strata(enum)enum=2   3.922e-01   0.057   0.9544
## as.factor(dukes)3:strata(enum)enum=2   4.628e-01  -0.993   0.3205
## as.factor(dukes)2:strata(enum)enum=3   5.065e-01  -0.871   0.3838
## as.factor(dukes)3:strata(enum)enum=3   5.717e-01  -0.706   0.4804
## as.factor(dukes)2:strata(enum)enum=4   7.173e-01   0.202   0.8396
## as.factor(dukes)3:strata(enum)enum=4   7.250e-01  -1.240   0.2150
## as.factor(dukes)2:strata(enum)enum=5   1.107e+00   0.481   0.6307
## as.factor(dukes)3:strata(enum)enum=5   1.110e+00  -0.292   0.7705
## as.factor(dukes)2:strata(enum)enum=6   7.376e-01 -20.998  < 2e-16
## as.factor(dukes)3:strata(enum)enum=6   1.023e+00  13.830  < 2e-16
## as.factor(dukes)2:strata(enum)enum=7   0.000e+00      NA       NA
## as.factor(dukes)3:strata(enum)enum=7   0.000e+00      NA       NA
## as.factor(dukes)2:strata(enum)enum=8   0.000e+00      NA       NA
## as.factor(dukes)3:strata(enum)enum=8   0.000e+00      NA       NA
## as.factor(dukes)2:strata(enum)enum=9   0.000e+00      NA       NA
## as.factor(dukes)3:strata(enum)enum=9   0.000e+00      NA       NA
## as.factor(dukes)2:strata(enum)enum=10  0.000e+00      NA       NA
## as.factor(dukes)3:strata(enum)enum=10  0.000e+00      NA       NA
## as.factor(dukes)2:strata(enum)enum=11  0.000e+00      NA       NA
## as.factor(dukes)3:strata(enum)enum=11  0.000e+00      NA       NA
## as.factor(dukes)2:strata(enum)enum=12  0.000e+00      NA       NA
## as.factor(dukes)3:strata(enum)enum=12  0.000e+00      NA       NA
## as.factor(dukes)2:strata(enum)enum=13  0.000e+00      NA       NA
## as.factor(dukes)3:strata(enum)enum=13  0.000e+00      NA       NA
```

```
## as.factor(dukes)2:strata(enum)enum=14  0.000e+00      NA       NA
## as.factor(dukes)3:strata(enum)enum=14  0.000e+00      NA       NA
## as.factor(dukes)2:strata(enum)enum=15  0.000e+00      NA       NA
## as.factor(dukes)3:strata(enum)enum=15  0.000e+00      NA       NA
## as.factor(dukes)2:strata(enum)enum=16  0.000e+00      NA       NA
## as.factor(dukes)3:strata(enum)enum=16  0.000e+00      NA       NA
## as.factor(dukes)2:strata(enum)enum=17  0.000e+00      NA       NA
## as.factor(dukes)3:strata(enum)enum=17  0.000e+00      NA       NA
## as.factor(dukes)2:strata(enum)enum=18  0.000e+00      NA       NA
## as.factor(dukes)3:strata(enum)enum=18  0.000e+00      NA       NA
## as.factor(dukes)2:strata(enum)enum=19  0.000e+00      NA       NA
## as.factor(dukes)3:strata(enum)enum=19  0.000e+00      NA       NA
## as.factor(dukes)2:strata(enum)enum=20  0.000e+00      NA       NA
## as.factor(dukes)3:strata(enum)enum=20  0.000e+00      NA       NA
## as.factor(dukes)2:strata(enum)enum=21  0.000e+00      NA       NA
## as.factor(dukes)3:strata(enum)enum=21  0.000e+00      NA       NA
## as.factor(dukes)2:strata(enum)enum=22  0.000e+00      NA       NA
## as.factor(dukes)3:strata(enum)enum=22  0.000e+00      NA       NA
## as.factor(dukes)2:strata(enum)enum=23  0.000e+00      NA       NA
## as.factor(dukes)3:strata(enum)enum=23  0.000e+00      NA       NA
##
## Likelihood ratio test=54.79  on 12 df, p=1.971e-07
## n= 598, number of events= 313
##    (263 observations deleted due to missingness)
```

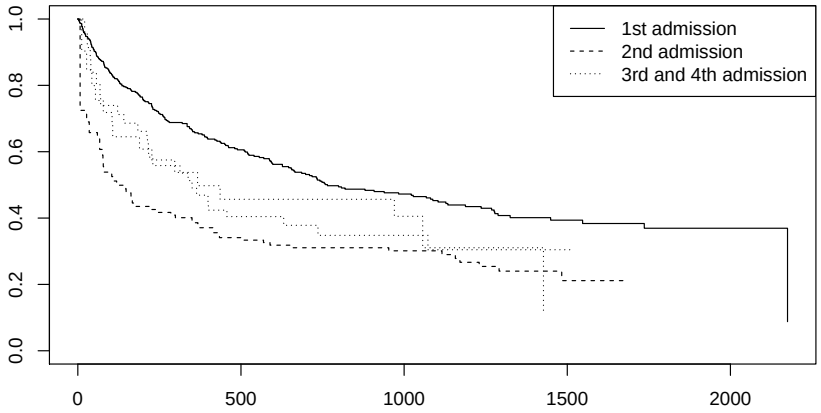
Notice that coefficients of some covariates cannot be estimated. If the number of subjects decreases as k (event) increases, stable coefficient estimates cannot be obtained for these higher ranks, k, of hospital admission. We have several possibilities to deal with this. The first is to aggregate, for example, the final stratas, which can be done by creation of a limited variable `enum5=min(enum,5)`. Another possibility is truncate the data set after the fifth hospital admission (`copd2.5 <- subset(copd, enum<5)`).

```
pwp.fit.strata.5 <- coxph(Surv(start, stop, event) ~ as.factor(dukes)* strata(enum) +
                  cluster(hc) , data=subset(colon2, enum<5))
pwp.fit.strata.5
## Call:
## coxph(formula = Surv(start, stop, event) ~ as.factor(dukes) +
##     strata(enum) + as.factor(dukes):strata(enum), data = subset(colon2,
##     enum < 5), cluster = hc)
##
##                                      coef exp(coef) se(coef) robust se
## as.factor(dukes)2                 0.39862   1.48977  0.15985   0.16129
## as.factor(dukes)3                 1.33973   3.81801  0.19537   0.17984
```

```
## as.factor(dukes)2:strata(enum)enum=2  0.02244   1.02269  0.40561   0.39216
## as.factor(dukes)3:strata(enum)enum=2 -0.45978   0.63143  0.50717   0.46282
## as.factor(dukes)2:strata(enum)enum=3 -0.44108   0.64334  0.50855   0.50646
## as.factor(dukes)3:strata(enum)enum=3 -0.40340   0.66804  0.56639   0.57167
## as.factor(dukes)2:strata(enum)enum=4  0.14519   1.15626  0.73746   0.71730
## as.factor(dukes)3:strata(enum)enum=4 -0.89891   0.40701  0.76451   0.72504
##                                          z        p
## as.factor(dukes)2                    2.471   0.0135
## as.factor(dukes)3                    7.450 9.36e-14
## as.factor(dukes)2:strata(enum)enum=2  0.057   0.9544
## as.factor(dukes)3:strata(enum)enum=2 -0.993   0.3205
## as.factor(dukes)2:strata(enum)enum=3 -0.871   0.3838
## as.factor(dukes)3:strata(enum)enum=3 -0.706   0.4804
## as.factor(dukes)2:strata(enum)enum=4  0.202   0.8396
## as.factor(dukes)3:strata(enum)enum=4 -1.240   0.2150
##
## Likelihood ratio test=48.79  on 8 df, p=6.963e-08
## n= 558, number of events= 285
##    (202 observations deleted due to missingness)
```

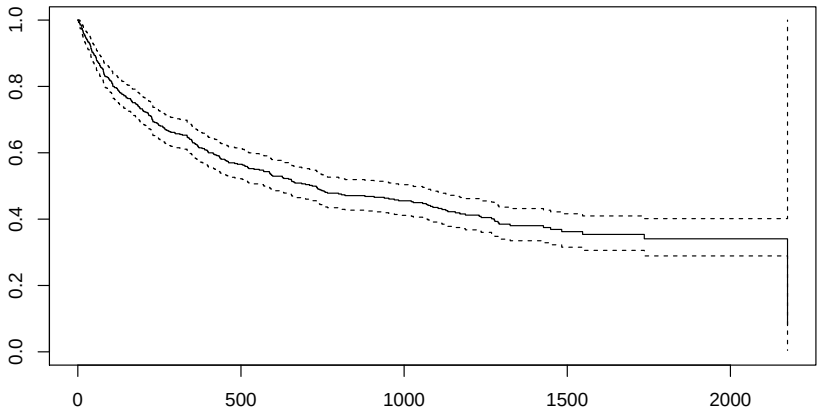
Baselines survival functions can be estimated and plotted using the function `survfit()`.

```
pwp.fit.strata <- coxph(Surv(start, stop, event) ~ as.factor(dukes)
                        + strata(enum) + cluster(hc),
                        data=subset(colon2, enum<5))
plot(survfit(pwp.fit.strata), lty=c(1,2,3,3))
legend("topright", c("1st admission", "2nd admission", "3rd and 4th admission"),
       lty=c(1,2,3))
```



Which is different of the baseline hazard function assumed in the AG model

```
ag.fit <- coxph(Surv(start, stop, event) ~ as.factor(dukes) +
                     cluster(hc),
                     data=subset(colon2, enum<5))
plot(survfit(ag.fit))
```



The Wei, Lin and Weisfeld (WLW) requires total time formulation but it is required to have the same number of entries for each individual. Let us assume that in this example we are considering 3 recurrences. The data should be:

```
hc    time start stop event enum
5634    24     0   24     1    1
```

```
5634   433     0  409     1    2
5634   580     0  147     0    3
10767  489     0  489     1    1
10767  693     0  204     0    2
10767  693     0  204     0    3
...
```

And the model is equally fitted by

```
wlw.fit.total <- coxph(Surv(time, event) ~ as.factor(dukes) + strata(enum) +
                  cluster(hc) , data=subset(colon2, enum<5))
```

Kelly and Lim review all these models in a paper by providing some examples of real data analysis and further discussion about the use of these models. The paper can be found in the repository.

5 Exercise (to deliver)

Data `lymphoma` is available at `gcmrec` package. It contains cancer relapses times after first treatment in patients diagnosed with low grade lymphoma. Data can be loaded into R by executing

```
data(lymphoma, package = "gcmrec")
```

NOTE: variable *time* contains inter-occurrence times, *event* is the censoring variable that is 1 for cancer relapses and 0 for the last follow up time indicating that the event is not observed and the variable *id* identifies each patient.

Exercise 1:

- Estimate the AG, PWP-Gap time, PWP-Total time and WLW models to determine whether there are differences in the relapsing time depending on the number of lesions at diagnosis (variable *distrib*). NOTE: you will need to create the proper data frames for PWT-Gap time and WLW models.
- Do we obtain the same conclusion by using the three models? (NOTE: use some of the functions we have seen in the lectures to prepare the required data)
- Repeat the same analyses adjusting for sex and response to the treatment (variable *tt effage*). Do we obtain the same conclusion as in the models without such adjustment?

NOTE: variable *distrib* encodes the lesions involved at diagnosis and has 4 categories (0=Single, 1=Localized, 2=More than one nodal site, 3=Generalized)

6 References

- The [`survival`] package (https://cran.r-project.org/web/packages/survival/)
- Andersen, P. K. and Gill, R. D. (1982). Cox's regression model for counting processes: a large sample study. Annals of Statistics, 10:1100-20.
- Prentice, R. L., Williams, B. J., and Peterson, A. V. (1981). On the regression analysis of multivariate failure time data. Biometrika, 68:373-389.
- Wei, L. J., Lin, D. Y., and Weissfeld, L. (1989). Regression analysis of multivariate incomplete failure time data by modeling marginal distributions. Journal of the American Statistical Association, 84:1065-1073.

- Kelly P.J., Lim L.L. (2000). Survival analysis for recurrent event data: an application to childhood infectious diseases. Statistics in Medicine, 19(1):13-33.

7 Session information

```
## R version 4.0.2 (2020-06-22)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 19042)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=Spanish_Spain.1252  LC_CTYPE=Spanish_Spain.1252
## [3] LC_MONETARY=Spanish_Spain.1252 LC_NUMERIC=C
## [5] LC_TIME=Spanish_Spain.1252
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] survival_3.2-3   knitr_1.29       BiocStyle_2.16.0
##
## loaded via a namespace (and not attached):
##  [1] bookdown_0.20       lattice_0.20-41     digest_0.6.25
##  [4] grid_4.0.2          magrittr_1.5        evaluate_0.14
##  [7] rlang_0.4.10        stringi_1.4.6       Matrix_1.2-18
## [10] rmarkdown_2.7       splines_4.0.2       tools_4.0.2
## [13] stringr_1.4.0       xfun_0.16           yaml_2.2.1
## [16] compiler_4.0.2      BiocManager_1.30.10 htmltools_0.5.0
```