Exercici d'events recurrents en R (part III)

Jose Calatayud Mateu

2025-05-11

```
## Llibreria
library(devtools)
library(tidyverse)
devtools::install_github("isglobal-brge/survrec",
                         build = FALSE)

library(survrec)
library(kableExtra)
```

Procedim a descarregar manualment l'arxius assosciats als paquet "gcmrec" que està dedicat a funcions de modelatge de models generals per events recurrents i el data sobre el qual treballarem "lymphoma":

```
source(file="lymphoma.R")
source(file="survrec.R")
source(file="gcmrec.R")
```

El data "lymphoma" contenen els temps de recaiguda del càncer després del primer tractamenten pacients diagnosticats amb lymphoma de grau baix.

```
head(lymphoma)
```

```
##   id      time event enum delay age sex distrib effage
## 1  6  3.900826     0    1    17  79   1       1     CR
## 2  7 63.173554     0    1    33  25   1       1     CR
## 3  8 41.289256     0    1    26  37   1       2     CR
## 4 11 29.421488     1    1    31  43   2       2     CR
## 5 11 20.826446     1    2    31  43   2       2     CR
## 6 11 17.950413     0    3    31  43   2       2     CR
```

NOTA: la variable *time* conté els temps entre esdeveniments, *event* és la variable de censura que val 1 per a recaigudes de càncer i 0 per al darrer moemnt de seguiment (indicant que l'esdeveniment no s'ha observat), i la variable *id* identifica cada pacient.

Exercici 2:

Estima un model frailty (amb el frailtypack package)per investigar si hi ha diferències en el risc de tindre recaigudes de cancer pel que fa al nombre de lesions en la variable de diagnostic (distrib) mitjançant els següents models:

```r
library(frailtypack)
```

```
## Loading required package: doBy

##
## Attaching package: 'doBy'

## The following object is masked from 'package:dplyr':
##
##     order_by

## Loading required package: MASS

##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select

## Loading required package: survC1

## Loading required package: survival

##
## Attaching package: 'survival'

## The following object is masked from 'package:boot':
##
##     aml

##
## Attaching package: 'frailtypack'

## The following object is masked from 'package:survival':
##
##     cluster
```

- *Gap inter-occurrence time scale and no effect of accumulating previous relapses*

Primer, tenim que tindre la base de dades en la estructura correcta del temps, com que el gap time scale i ho obtenim amb la següent funció:

```r
# 1. Ordenem la base de dades per id i temps
lymphoma <- lymphoma[order(lymphoma$id, lymphoma$time), ]

# 2. Transformem les dades incorpornant carlendar scale
lymphoma <- lymphoma %>%
  group_by(id) %>%
  mutate("t.start" = lag(cumsum(time), default = 0)) %>%
  mutate("t.stop" = t.start + time) %>%
  ungroup()
```

Mostrem el database lymphoma amb l'escala de gap time desitjada:

```
head(lymphoma)
```

```
## # A tibble: 6 x 11
##      id  time event  enum delay   age   sex distrib effage t.start t.stop
##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>   <dbl> <chr>    <dbl>  <dbl>
## 1     6  3.90     0     1    17    79     1       1 CR         0     3.90
## 2     7 63.2      0     1    33    25     1       1 CR         0    63.2
## 3     8 41.3      0     1    26    37     1       2 CR         0    41.3
## 4    11 18.0      0     3    31    43     2       2 CR         0    18.0
## 5    11 20.8      1     2    31    43     2       2 CR        18.0  38.8
## 6    11 29.4      1     1    31    43     2       2 CR        38.8  68.2
```

Fem proves amb la funció frailtyPenal(), veiem que dona problemes, existeixen time amb valor 0. Aleshores, tenim que filtrar tots aquells valors que tenen un time nul, en efecte:

```
lymphoma <- lymphoma %>% filter( time != 0 )
```

Ara passem a modelar tlal i com se'ns demana en l'enunciat:

1. *Gap inter-occurrence time scale and no effect of accumulating previous relapses*

```
# Model 1: Gap time, sense efecte acumulatiu
model1 <- frailtyPenal(
  Surv(time= time,  event) ~ cluster(id) + as.factor(distrib),
  data = lymphoma, n.knots = 8, recurrentAG = FALSE, kappa=10000,
  cross.validation = TRUE)
```

```
##
## Be patient. The program is computing ...
## The program took 0.09 seconds
```

```
model1
```

```
## Call:
## frailtyPenal(formula = Surv(time = time, event) ~ cluster(id) +
##     as.factor(distrib), data = lymphoma, recurrentAG = FALSE,
##     cross.validation = TRUE, n.knots = 8, kappa = 10000)
##
##
##   Shared Gamma Frailty model parameter estimates
##   using a Penalized Likelihood on the hazard function
##
##              coef exp(coef) SE coef (H) SE coef (HIH)       z        p
## distrib1 0.860248   2.36375    0.498117      0.498117 1.72700 0.084168
## distrib2 1.185762   3.27318    0.507274      0.507274 2.33752 0.019412
## distrib3 0.876820   2.40325    0.735146      0.735146 1.19272 0.232980
##
##          chisq df global p
## distrib 5.5078  3    0.138
```

```
##
##     Frailty parameter, Theta: 1.45385e-10 (SE (H): 1.15996e-05 ) p = 0.49999
##
##       penalized marginal log-likelihood = -220.56
##       Convergence criteria:
##       parameters = 0.000131 likelihood = 0.000588 gradient = 2.28e-07
##
##       LCV = the approximate likelihood cross-validation criterion
##             in the semi parametrical case     = 2.1324
##
##       n= 110
##       n events= 49  n groups= 63
##       number of iterations:  5
##
##       Exact number of knots used:  8
##       Best smoothing parameter estimated by
##       an approximated Cross validation:  0.187682, DoF:  8.00
```

2. *Calendar inter-occurrence time scale and no effect of accumulating previous relapses*

```
# Model 2: Calendar time, sense efecte acumulatiu
model2 <- frailtyPenal(
  Surv(time= t.start, time2 = t.stop,  event) ~ cluster(id) + as.factor(distrib),
  data = lymphoma, n.knots = 8, recurrentAG = FALSE, kappa=10000,
  cross.validation = TRUE)
model3
```

3. *Gap time inter-occurrence time scale and effect of accumulating previous relapses*

```
# Model 3: Gap time, amb efecte acumulatiu
model3 <- frailtyPenal(
  Surv(time= time,  event) ~ cluster(id) + as.factor(distrib) + enum,
  data = lymphoma, n.knots = 8, recurrentAG = FALSE, kappa=10000,
  cross.validation = TRUE)
```

```
##
## Be patient. The program is computing ...
## The program took 0.07 seconds
```

```
model3
```

```
## Call:
## frailtyPenal(formula = Surv(time = time, event) ~ cluster(id) +
##     as.factor(distrib) + enum, data = lymphoma, recurrentAG = FALSE,
##     cross.validation = TRUE, n.knots = 8, kappa = 10000)
##
##
##   Shared Gamma Frailty model parameter estimates
##   using a Penalized Likelihood on the hazard function
##
##               coef exp(coef) SE coef (H) SE coef (HIH)         z        p
## distrib1  0.910127  2.484639    0.497228      0.497228  1.830402 0.067190
```

```
## distrib2  1.246787  3.479147    0.495890      0.495890  2.514242 0.011929
## distrib3  1.034209  2.812881    0.769168      0.769168  1.344583 0.178760
## enum     -0.103577  0.901606    0.166806      0.166806 -0.620942 0.534640
##
##           chisq df global p
## distrib 6.33485  3   0.0964
##
##     Frailty parameter, Theta: 2.89553e-05 (SE (H): 0.0418774 ) p = 0.49972
##
##       penalized marginal log-likelihood = -220.37
##       Convergence criteria:
##       parameters = 4.88e-05 likelihood = 0.000168 gradient = 3.44e-07
##
##       LCV = the approximate likelihood cross-validation criterion
##             in the semi parametrical case     = 2.1397
##
##       n= 110
##       n events= 49  n groups= 63
##       number of iterations:  6
##
##       Exact number of knots used:  8
##       Best smoothing parameter estimated by
##       an approximated Cross validation:  0.187682, DoF:  8.00
```

4. *Calendar inter-occurrence time scale and effect of accumulating previous relapses*

```r
# Model 4: Calendar time, amb efecte acumulatiu
model2 <- frailtyPenal(
  Surv(time= t.start, time2 = t.stop,  event) ~ cluster(id) + as.factor(distrib) + enum,
  data = lymphoma, n.knots = 8, recurrentAG = FALSE, kappa=10000,
  cross.validation = TRUE)
model4
```

6. *Provide an interpretation of model parameters of this last model*

No podem aporta una interpretació dels paràmetres de l'últim model perque el model proposat en el apartat 5 del docuement no cal realitzar-ho per a l'exercici 2

7. *Which is the most adequate model? Why? (HINT: use the Akaike criteria since the {gcmrec} function is providing likelihood of each model)*

Els únics models que podem comporar son el model1 i model3 amb escala temps gap-time, que son els únics en que la funció frailtyPenal ha funcionat. Un forma per comparar seria utilitzant la sortida de la funció del model amb argument AIC però en ambdos casos presenta un AIC 0, ja que els criteris clàssics no són aplicables de manera directa als models penalitzats.

És correcte utilitzar el logLikPenal per calcular AIC i BIC? No del tot, els criteris AIC i BIC clàssics es basen en la log-versemblança no penalitzada, i les fórmules són:

$$\text{AIC} = -2 \cdot \log L + 2k \text{BIC} = -2 \cdot \log L + k \cdot \log(n)$$

On:

- $\log L$: log-versemblança del model (no penalitzada)
- k: nombre de paràmetres estimats
- n: nombre d'observacions

Tanmateix, `frailtyPenal()` utilitza una log-versemblança penalitzada, que ja inclou termes per regularització (com ara la suavitat dels splines). Per això, els valors d'AIC i BIC no són directament interpretables com en models no penalitzats.

Quan sí que té sentit? Seria raonable utilitzar `logLikPenal` per comparar models del mateix tipus de forma heurística, estimats amb el mateix nivell de penalització. Això és útil, per exemple, quan es volen comparar diferents nombres de splines o especificacions del model amb la mateixa estructura.

Llavors, passem a calcular de forma manual el valor del AIC i el BIC en els dos models per poder-los comparar:

```
# model1
logLik1 <- model1$logLikPenal
k1 <- model1$npar
n <- model1$n

# model3
logLik3 <- model3$logLikPenal
k3 <- model3$npar

# Càlculs
AIC1 <- -2 * logLik1 + 2 * k1
BIC1 <- -2 * logLik1 + k1 * log(n)

AIC3 <- -2 * logLik3 + 2 * k3
BIC3 <- -2 * logLik3 + k3 * log(n)

cat("Model 1: AIC =", AIC1, " BIC =", BIC1, "\n")
```

```
## Model 1: AIC = 469.1281  BIC = 506.9348
```

```
cat("Model 3: AIC =", AIC3, " BIC =", BIC3, "\n")
```

```
## Model 3: AIC = 470.7349  BIC = 511.2421
```

Aleshores, el model que presenta un AIC inferior és el model1, doncs aquest seria el nostre model desitjat dintre dels dos. Encara aue el valors son molt ajustat i no hi ha gaire diferències entre escollir uno i un altre, endemés en el model3 el coefients associat al terme acumulatiu no és significatiu, aixina que poptser el model1 sí que és el adequat donat aquest conttext

Encara que l'AIC i el BIC clàssics no són aplicables directament als models penalitzats, és acceptable calcular-los manualment utilitzant `logLikPenal` si:

- Es comparen models amb la mateixa estructura i tipus de penalització.
- L'objectiu és seleccionar el model més adequat dins d'una mateixa família.

En cas contrari, s'aconsella utilitzar mètriques específiques del paquet o tècniques de validació creuada per fer comparacions més robustes. No ens serveix el calcúl del log-likelihood penalitzat per fer comparacions ja que no reporta el fet de que el model3 incorpora un variable més que fa que el log-likelihood sigui superior però no es comparable degut a que incorpora una variable més.