Primer acercamiento Cox Bayesiano

Analizamos breve del código expuesto en el artículo Bayesian Survival Analysis with BUGS de las páginas 7-9.

Cargamos las librerías necesarias, entre ellas "KMsurv" que contiene el marco de datos "larynx". Este marco de datos contiene las variables:

- Stage (categórica de 4 niveles, 1-4)
- Age (de tipo entero)
- Time (continua)
- Diagyr (de tipo entero)
- Delta(binaria, 0-1. Nos indica si hay censura/falla, la censura es 1)

Recordemos que en el modelo de Cox no se considera el intersecto así, las covariables que se asocian a la parte de la región lineal serán 6. A saber estas son, los estapas 2-3 de la enfermedad (Stage) y las covariables asociadas a la edad (Age) y el año de dignóstico de cancer de laringe(Diagyr).

Es decir, para el modelo de Cox concideramos una regresión (que después ligamos a una exponencial) del tipo $\beta_2 I_{\text{stage }=\ 2} + \beta_3 I_{\text{ stage }=\ 3} + \beta_3 I_{\text{ stage }=\ 3} + \beta_4 \text{Age} + \beta_5 \text{Diagyr}$

```r
library(rjags)
library(coda)
library(KMsurv)
library(dplyr)
library(tidyverse)
data(larynx) #importamos marco de datos
X <- larynx #marco de datos para definir la matriz diseño
time <- larynx$time #definfimos un vector con las observaciónes de la variable "time"
```

Construimos una sucesión aritmética para definir una partición de la recta en la que caen las observaciones. Escogemos clasificar las observaciones del timpo de supervivencia en $K = 3$ itervalos. Para ello se construye una matriz auxiliar $D = \left[I_{(a_{k-1},a_k]}(t_i)\right]_{i,k}$ que nos indica si la observación i del tiempo de supervivencia cae en el k-ésimo intervalo. Usamos después esta matriz para contruir un vector que los indica en que intervalo cae la obsrvación i-ésima del tiempo de supervivencia t.

```r
time <- larynx$time #definfimos un vector con las observaciónes de la variable "time"
K <- 3 #número de intervalo que escogemos
X <- larynx #marco de datos
a <- seq(0, max(larynx$time) + 0.001, length.out = K + 1) #vector de tiempos de censura
int.obs <- matrix(data = NA, nrow = nrow(larynx), ncol = length(a) - 1)
D <- matrix(data = NA, nrow = nrow(larynx), ncol = length(a) - 1)
for(i in 1:nrow(larynx)){
```

```r
  for (k in 1:(length(a) - 1)){
    D[i, k] <- ifelse(time[i] - a[k] > 0, 1, 0) * ifelse(a[k + 1] - time[i] > 0, 1, 0)#indicadora del i
    int.obs[i, k] <- D[i, k] * k #indica en que intervalo (1,2,3) cae la observación.
  }
}
int.obs <- rowSums(int.obs) #vector que nos indica en que intervalo cae cada observación
```

Declaramos los hiperparámetros e inciamos la simulación de las ditribuciones a posteriori con valores simulados.

```r
#datos con los cuales vamos a entrenar el modelo
data.jags <- list(n = nrow(larynx), m=length(a)-1, a = a,
                  delta=larynx$delta, time=larynx$time, X=X[,-1],
                  int.obs=int.obs, Nbetas=ncol(X)-1, zeros=rep(0,nrow(larynx)))

#función para inicializar el modelo
init.jags <- function(){list(beta = rnorm(ncol(X)-1), lambda =runif(3,0.1))}
#parámetros que vamos a monitorear
param.jags <-c("beta", "lambda")  #paramteros a monitorear
```

Sobre el código implementado en BUGS

Sean $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_n)$, $\boldsymbol{x} = (x, \ldots, x_n)$. El modelo de regresión de Cox semiparamétrico considera que:

- $h(t_i|\theta) = (\sum_{m=1}^{K} \lambda_m I_{(a_{m-1},a_m]}(t_i))e^{\boldsymbol{\beta}^T \boldsymbol{x}}$
- $L(\theta|\boldsymbol{x}) = \prod_m h(t_i|\theta)^{\delta_m} S(t_i|\theta)$

La siguiente cadena de caracteres nos muestra el contenido del archivo .txt con la descripción del modelo. Analicemos este pedazo de código:

```r
descrip_txt <-"model{
  for(i in 1:n){
    for(k in 1:int.obs[i]){
      cond[i,k] <- step(time[i] - a[k+1])
      HH[i,k] <- cond[i,k]*(a[k+1]-a[k])*lambda[k] +
                  (1-cond[i,k])*(time[i]-a[k])*lambda[k]
    }
    #fun de riesgo base acumulado
    H[i] <- sum(HH[i,1:int.obs[i]])
  }
  for(i in 1:n){
    #predictor lineal
    elinpred[i] <- exp(inprod(beta[],X[i,]))

    #fun de logriesgo
    logHaz[i] <- log(lambda[int.obs[i]]*elinpred[i])

    #fun de logsupervivencia
    logSurv[i] <- -H[i]*elinpred[i]

    #logverosimilitud (usando el truco de los ceros)
    phi[i] <- 100000 - delta[i]*logHaz[i]-logSurv[i]
    zeros[i] ~ dpois(phi[i])
  }
```

```
  #priors lambda y beta
  for(l in 1:Nbetas){beta[l] ~ dnorm(0,0.001)}
  for(r in 1:m){lambda[r] ~ dgamma(0.01,0.01)}
}"
```

Compilado del modelo e inferencia.

```
#compilación del modelo
Modelo_compilado <- jags.model(data = data.jags,file = "modelo_prueba1.txt",
                               inits = init.jags, n.chains = 3)
```

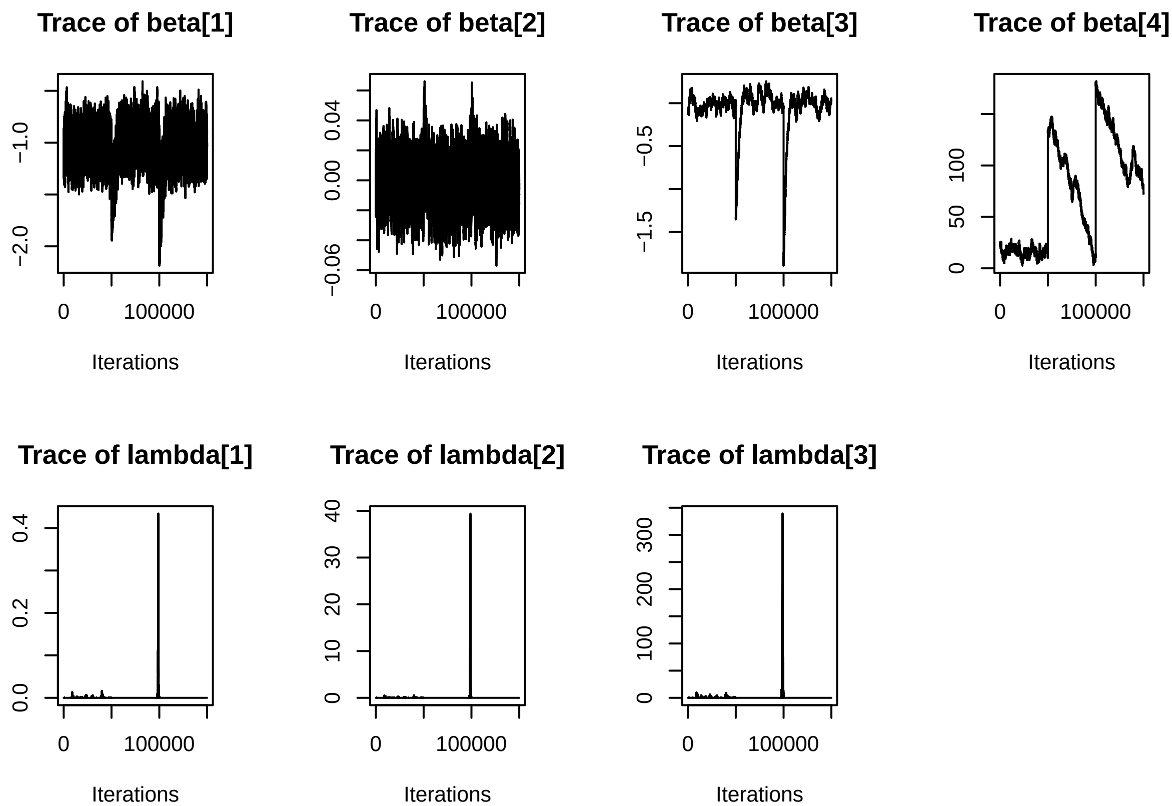
```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 90
##    Unobserved stochastic nodes: 7
##    Total graph size: 2182
##
## Initializing model
```

```
#Mandamos llamar coda para tomar muestras para dist a posteriori
update(Modelo_compilado, 1000)
res <- coda.samples(Modelo_compilado,variable.names=param.jags,n.iter=50000, n.thin=10)
```

```
#unimos las 3 cadenas MCMC para hacer inferencia sobre los resultados de la simulación de
#las dist posteriores
results <- as.mcmc(do.call(rbind,res))
```

Graficamos las trazas de las simulaciones para los hiperparámetros

```
#Trazas
par(mfrow=c(2,4))
traceplot(results)
```



Obtenemos información sobre las distribuciones a posteriori

```
#Info sobre dist posteriores
summary(results)
```

```
##
## Iterations = 1:150000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 150000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                 Mean        SD  Naive SE Time-series SE
## beta[1]   -1.0308357  0.177274 4.577e-04      6.028e-03
## beta[2]   -0.0018110  0.014009 3.617e-05      3.797e-04
## beta[3]   -0.0471078  0.248359 6.413e-04      4.977e-02
## beta[4]   72.9252068 51.750116 1.336e-01      2.370e+01
## lambda[1]  0.0005328  0.008038 2.075e-05      4.653e-04
## lambda[2]  0.0217310  0.359452 9.281e-04      1.159e-02
## lambda[3]  0.3839671  4.981086 1.286e-02      1.980e-01
##
## 2. Quantiles for each variable:
##
##                 2.5%        25%        50%        75%      97.5%
## beta[1]   -1.479e+00 -1.120e+00 -1.015e+00 -9.158e-01  -0.737597
## beta[2]   -2.833e-02 -1.119e-02 -2.329e-03  7.070e-03   0.027168
## beta[3]   -9.133e-01 -6.551e-02  4.397e-03  5.443e-02   0.177726
```

```
## beta[4]     7.922e+00  1.831e+01  8.194e+01  1.139e+02 164.344744
## lambda[1]  3.565e-66  7.507e-48  2.153e-35  1.624e-07   0.001469
## lambda[2]  1.612e-64  3.722e-46  1.161e-33  7.911e-06   0.071869
## lambda[3]  3.715e-63  8.678e-45  3.089e-32  1.761e-04   1.822404
```