This Rmarkdown document helps user to buid and fit joint models using the code extension available on Github. This document relies on the examples presented in the article.

Prothrombin example

We consider the data available in package JM: the prothro data set. 488 liver cirrhosis patients are followed at most 12 days with prothrombin measurements. Status (dead/censored) at the end of the follow-up is available for each patient. The objective is to assess the link between the individual prothrombin evolution with the risk of death.

```
library(JM)
```

```
## Warning: package 'JM' was built under R version 4.0.5
```

```
## Loading required package: MASS
```

```
## Loading required package: nlme
```

```
## Warning: package 'nlme' was built under R version 4.0.5
```

```
## Loading required package: splines
```

```
## Loading required package: survival
```

```
## Warning: package 'survival' was built under R version 4.0.5
```

```
library(pracma)
```

```
## Warning: package 'pracma' was built under R version 4.0.5
```

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.0.5
```

```
library(Cairo)
library(viridis)
```

```
## Warning: package 'viridis' was built under R version 4.0.5
```

```
## Loading required package: viridisLite
```

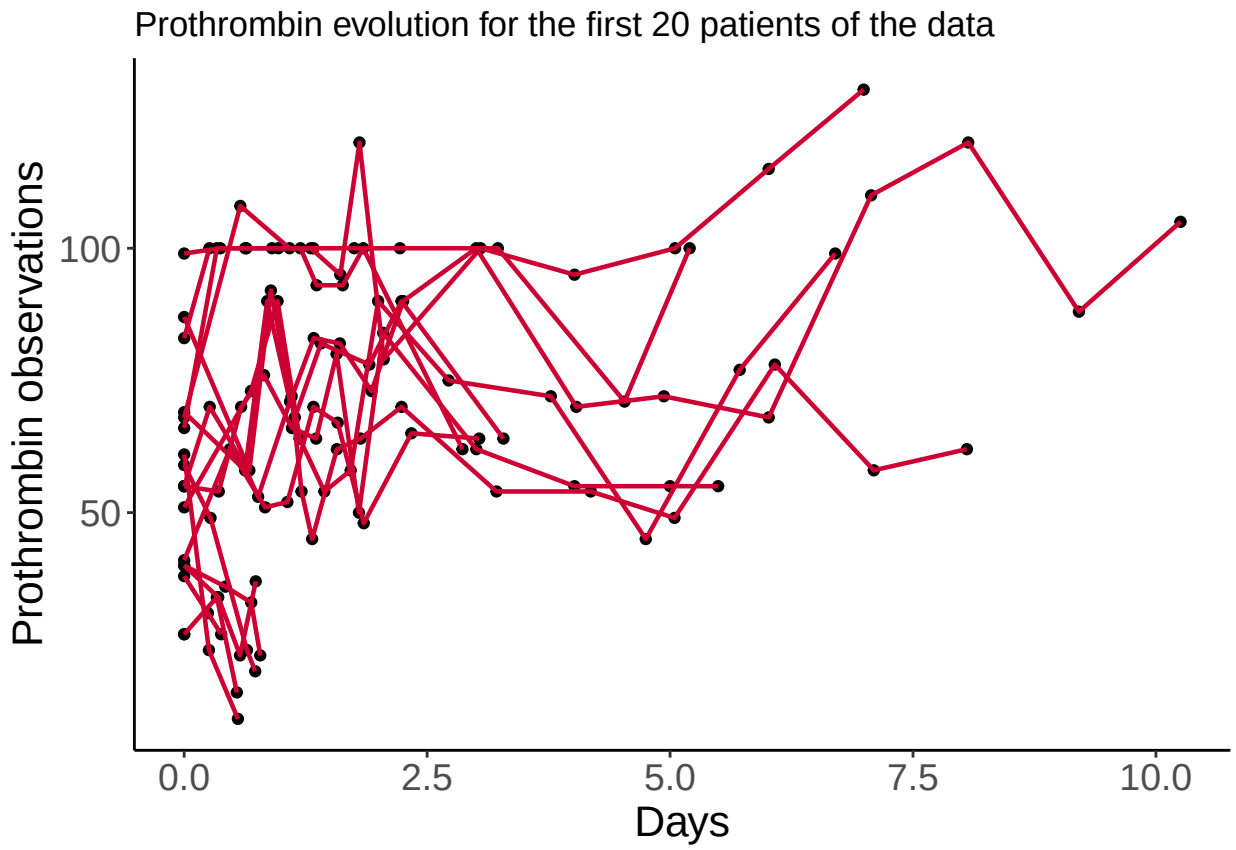
```
## Warning: package 'viridisLite' was built under R version 4.0.5
```

```
library(rlang)
```

```
## Warning: package 'rlang' was built under R version 4.0.5
```

```
data("prothro")
data("prothros")

gp = ggplot(data=prothro[which(prothro$id %in% 1:20),], aes(x=time, y=pro, group = id))+
  geom_point(lwd=1.5)+geom_line(col="#CC0033",lwd=0.8)+theme_classic()+
  ylab("Prothrombin observations")+xlab("Days")+
  theme(axis.text = element_text(size=14),axis.title = element_text(size=16))+
  ggtitle(label = "Prothrombin evolution for the first 20 patients of the data")
gp
```




```
table(prothros$death)
```

```
##
##   0   1
## 196 292
```

Joint model fit using new saemix.multi() function

The extended code uses the same main functions as saemix package does. We therefore refer the user to the saemix documentation previously published for the detail of each function (see Comets et al. JSS, 2017). Briefly, the main function is the saemix.multi() function used to estimate the population parameters of the (joint) model. This function requires two mandatory arguments referring to (1) the model (saemixModel object) and the data (saemixData object). The third argument is optional and concerns the algorithm settings.

```
saemixDir <- "C:/Users/AlexandraLAVALLEY/Documents/GitHub/saemixextension"
workDir <- file.path(saemixDir, "joint")
progDir<-file.path(saemixDir, "R")
setwd(workDir)

source(file.path(progDir,"aaa_generics.R"))
```

Loading functions from Github

```
## Creating a new generic function for 'psi' in the global environment

## Creating a new generic function for 'eta' in the global environment
```

```
source(file.path(progDir,"SaemixData.R"))
source(file.path(progDir,"SaemixRes.R"))
source(file.path(progDir,"SaemixModel.R"))
source(file.path(progDir,"SaemixObject.R"))
source(file.path(progDir,"func_plots.R"))

source(file.path(workDir,"multi_aux.R"))
source(file.path(workDir,"multi_initializeMainAlgo.R"))
source(file.path(workDir,"multi_estep.R"))
source(file.path(workDir,"multi_mstep.R"))
source(file.path(workDir,"multi_main.R"))
source(file.path(workDir,"multi_map.R"))
source(file.path(workDir,"compute_LL_multi.R"))
```

Formatting data The function saemixData() requires a mandatory argument which is the name of the dataset. The dataset has to be formatted in order to obtain an id column corresponding to the patient id, a time column corresponding to the sampling times, an observation column corresponding to the response observations and a ytype column corresponding to the distinct response types. Optional columns can be added if user the wants to model covariates for example. See saemix documentation for more details. In this example, prothrombin measurements correspond to response 1 (ytype = 1) and survival measurements correspond to response 2 (ytype = 2). For the survival reponse (ytype = 2), observation is 1 in case of event (death) and 0 otherwise.

```
d1 = prothro[,c(1,2,3)]
d1$ytype=1
colnames(d1)[2] = "obs"
d2 = prothros[,c(1,3,2)]
d2$ytype = 2
colnames(d2)[2] = "obs"
colnames(d2)[3] = "time"
data_joint = rbind(d1,d2)
# To see the data for patient 1
data_joint[data_joint$id==1,]
```

```
##       id obs      time ytype
## 1      1  38 0.0000000     1
```

```
## 2      1  31 0.2436754     1
## 3      1  27 0.3805717     1
## 11000  1   1 0.4134268     2
```

The user is encouraged to specify optional arguments of the saemixData() function: the id variable (name.group argument), the predictor variables (name.predictors argument) with at least the sampling times, the observation variable (name.reponse argument) and the response type variable (name.ytype argument).

```
saemix.data<-saemixData(name.data=data_joint, name.group=c("id"),
                        name.predictors=c("time","obs"),
                        name.response="obs",name.ytype = "ytype")
```

```
##
##
## The following SaemixData object was successfully created:
##
## Object of class SaemixData
##     longitudinal data for use with the SAEM algorithm
## Dataset data_joint
##     Structured data: obs ~ time + obs | id
##     X variable for graphs: time ()
```

Joint model with a linear mixed-effects model and a survival model with constant baseline hazard The saemixModel() function requires two mandatory arguments. The first one is a R function describing the joint model involving the structural model for longitudinal observations and the likelihood contribution for the survival observations. The second one is a matrix with a number of columns equal to the number of parameters, and one (when no covariates) or several row (when covariates enter the model) giving the initial estimates of fixed-effects. The user is encouraged to specify optional arguments of the saemixModel() function: the response type with the modeltype argument ("structural" for longitudinal observations and "likelihood" for survival ones), the distribution of each parameter with the transform.par argument (0 = normal, 1 = log-normal, 2 = probit and 3 = logit), the fixed or estimated parameters with the fixed.estim argument (0 = to be fixed to the initial estimate, 1 = to be estimated), if random effects are added with the covariance.model argument (square matrix of size equal to the number of parameters giving the variance-covariance matrix of the model), the initialization of random effect variances with the omega.init argument (square matrix of size equal to the number of parameters giving the initialization of the variance-covariance matrix of the model), the error model with the error.model argument (valid types are "constant", "proportional", "combined"). Futher arguments can be considered and found in the package description (see documentation).

In the following we start with a simple case: a joint model with a linear mixed-effects and a survival model involving constant baseline risk. The joint model writes:

$$\begin{aligned} y_{ij} =& m_l(t_{ij},\psi_i) + g[m_l(t_{ij},\psi_i),\sigma]\epsilon_{ij} \\ =& \psi_{i0} + \psi_{i1} \times t_{ij} + \sigma\epsilon_{ij} \\ h_i(t,\psi_i) =& h_0 \, \exp(\alpha \, m(t,\psi_i)) \end{aligned} \tag{1}$$

We then define the model to be entered in the function. This function must have 3 arguments named psi (assumed to be a matrix with the number of columns equal to the number of parameters in the model (excluding error parameters), so here 4 for μ_0, μ_1, h_0 and α), id (assumed to be a vector of indices matching observation number with subject index) and xidep (assumed to be a matrix with as many columns as

predictors + 1 for the type of response, so here 3 for time, observations and response type). The three arguments passed to the function will be generated automatically from the model and data object within the saemix code. The function must return a vector composed of predictions for longitudinal responses and likelihood contributions for the survival responses. The length of the vector is equal to the number of rows in the predictor xidep.

```
JMmodel<-function(psi,id,xidep) {
  ytype<-xidep$ytype  # type of response (1: continuous, 2: event)
  b0 <- psi[id,1]
  b1 <- psi[id,2]
  h0 <- psi[id,3]
  alpha <- psi[id,4]

  ypred <- b0+b1*xidep$time # predictions for the longitudinal part

  T<-xidep$time[ytype==2] # vector of times (survival response)
  Nj <- length(T)
  ev = xidep$obs[ytype==2] # vector of observations (survival response)
  cens<-which(ev==0)       # with censored ones
  ind <- which(ev==1)      # and event ones

  # Creating vectors of the same length of T to compute likelihood of the survival part
  #(so removing duplicates)
  b0b = b0[ytype==2] # to have vectors of the same length as T
  b1b = b1[ytype==2]
  h0b = h0[ytype==2]
  alphab = alpha[ytype==2]

  haz <- h0b*exp(alphab*(b0b+b1b*T)) # instantaneous hazard
  # cumulative hazard (explicit expression in that case)
  H <- (h0b/(alphab*b1b))*exp((b0b+b1b*T)*alphab)-(h0b/(alphab*b1b))*exp(alphab*b0b)

  logpdf <- rep(0,Nj)
  logpdf[cens] <- -H[cens] # likelihood contributions for censored observations
  logpdf[ind] <- -H[ind] + log(haz[ind]) # likelihood contributions for event observations

  ypred[ytype==2] = logpdf
  return(ypred)
}
```

```
#### initializing parameters

param<-c(73,1.25,0.6,0.0001)
omega.sim<-c(18, 3, 0.05, 0.01)
sigma.sim <- 17

### saemix Model

saemix.model<-saemixModel(model=JMmodel,description="JM LMEM-TTE constant baseline hazard",
                      modeltype=c("structural","likelihood"),
                      psi0=matrix(param,ncol=4,byrow=TRUE,
                                  dimnames=list(NULL, c("b0","b1","h0","alpha"))),
                      transform.par=c(0,0,1,0), covariance.model=diag(c(1,1,0,0)),
                      fixed.estim = c(1,1,1,1),error.model = "constant",
```

```
                        omega.init = diag(omega.sim))
```

```
##
##
## The following SaemixModel object was successfully created:
##
## Nonlinear mixed-effects model
##   Model function:  JM LMEM-TTE constant baseline hazard
##   Model type:  structural likelihood
## function(psi,id,xidep) {
##   ytype<-xidep$ytype  # type of response (1: continuous, 2: event)
##   b0 <- psi[id,1]
##   b1 <- psi[id,2]
##   h0 <- psi[id,3]
##   alpha <- psi[id,4]
##
##   ypred <- b0+b1*xidep$time # predictions for the longitudinal part
##
##   T<-xidep$time[ytype==2] # vector of times (survival response)
##   Nj <- length(T)
##   ev = xidep$obs[ytype==2] # vector of observations (survival response)
##   cens<-which(ev==0)       # with censored ones
##   ind <- which(ev==1)      # and event ones
##
##   # Creating vectors of the same length of T to compute likelihood of the survival part
##   #(so removing duplicates)
##   b0b = b0[ytype==2] # to have vectors of the same length as T
##   b1b = b1[ytype==2]
##   h0b = h0[ytype==2]
##   alphab = alpha[ytype==2]
##
##   haz <- h0b*exp(alphab*(b0b+b1b*T)) # instantaneous hazard
##   # cumulative hazard (explicit expression in that case)
##   H <- (h0b/(alphab*b1b))*exp((b0b+b1b*T)*alphab)-(h0b/(alphab*b1b))*exp(alphab*b0b)
##
##   logpdf <- rep(0,Nj)
##   logpdf[cens] <- -H[cens] # likelihood contributions for censored observations
##   logpdf[ind] <- -H[ind] + log(haz[ind]) # likelihood contributions for event observations
##
##   ypred[ytype==2] = logpdf
##   return(ypred)
## }
##   Nb of parameters: 4
##       parameter names:  b0 b1 h0 alpha
##       distribution:
##      Parameter Distribution Estimated
## [1,] b0        normal       Estimated
## [2,] b1        normal       Estimated
## [3,] h0        log-normal   Estimated
## [4,] alpha     normal       Estimated
##   Variance-covariance matrix:
##       b0 b1 h0 alpha
## b0     1  0  0     0
```

```
## b1     0  1  0     0
## h0     0  0  0     0
## alpha  0  0  0     0
##   Error model: constant , initial values: a.1=1
##     No covariate in the model.
##     Initial values
##              b0   b1  h0 alpha
## Pop.CondInit 73 1.25 0.6 1e-04
```

In the following we specify some algorithm settings. The option fim = T is specified to obtain standard errors of parameter estimates. ll.is is specified to obtain the loglikelihood at the MLE, the AIC and BIC. Graphs are not currently adapted so please specify save.graphs = F. We run the algorithm using the saemix.multi() function.

```
saemix.options<-saemixControl(seed=12345, map=T, fim=T, ll.is=TRUE, save.graphs = F)
# please, specify save.graphs=F (currently not extended)
yfit <- saemix.multi(saemix.model, saemix.data, saemix.options)
```

```
## Nonlinear mixed-effects model fit by the SAEM algorithm
## -----------------------------------
## ----          Data             ----
## -----------------------------------
## Object of class SaemixData
##     longitudinal data for use with the SAEM algorithm
## Dataset data_joint
##     Structured data: obs ~ time + obs | id
##     X variable for graphs: time ()
## Dataset characteristics:
##     number of subjects:     488
##     number of observations: 3456
##     average/min/max nb obs: 7.08  /  2  /  18
## First 10 lines of data:
##       id      time obs obs.1 mdv cens occ ytype
## 1      1 0.0000000  38    38   0    0   1     1
## 2      1 0.2436754  31    31   0    0   1     1
## 3      1 0.3805717  27    27   0    0   1     1
## 11000  1 0.4134268   1     1   0    0   1     2
## 4      2 0.0000000  51    51   0    0   1     1
## 5      2 0.6872194  73    73   0    0   1     1
## 6      2 0.9610119  90    90   0    0   1     1
## 7      2 1.1882598  64    64   0    0   1     1
## 8      2 1.4428869  54    54   0    0   1     1
## 9      2 1.7139415  58    58   0    0   1     1
## -----------------------------------
## ----          Model            ----
## -----------------------------------
## Nonlinear mixed-effects model
##   Model function:  JM LMEM-TTE constant baseline hazard
##   Model type:  structural likelihood
## function(psi,id,xidep) {
##   ytype<-xidep$ytype  # type of response (1: continuous, 2: event)
##   b0 <- psi[id,1]
##   b1 <- psi[id,2]
```

```
##   h0 <- psi[id,3]
##   alpha <- psi[id,4]
##
##   ypred <- b0+b1*xidep$time # predictions for the longitudinal part
##
##   T<-xidep$time[ytype==2] # vector of times (survival response)
##   Nj <- length(T)
##   ev = xidep$obs[ytype==2] # vector of observations (survival response)
##   cens<-which(ev==0)       # with censored ones
##   ind <- which(ev==1)      # and event ones
##
##   # Creating vectors of the same length of T to compute likelihood of the survival part
##   #(so removing duplicates)
##   b0b = b0[ytype==2] # to have vectors of the same length as T
##   b1b = b1[ytype==2]
##   h0b = h0[ytype==2]
##   alphab = alpha[ytype==2]
##
##   haz <- h0b*exp(alphab*(b0b+b1b*T)) # instantaneous hazard
##   # cumulative hazard (explicit expression in that case)
##   H <- (h0b/(alphab*b1b))*exp((b0b+b1b*T)*alphab)-(h0b/(alphab*b1b))*exp(alphab*b0b)
##
##   logpdf <- rep(0,Nj)
##   logpdf[cens] <- -H[cens] # likelihood contributions for censored observations
##   logpdf[ind] <- -H[ind] + log(haz[ind]) # likelihood contributions for event observations
##
##   ypred[ytype==2] = logpdf
##   return(ypred)
## }
## <bytecode: 0x000000002469d848>
##   Nb of parameters: 4
##       parameter names:  b0 b1 h0 alpha
##       distribution:
##      Parameter Distribution Estimated
## [1,] b0        normal       Estimated
## [2,] b1        normal       Estimated
## [3,] h0        log-normal   Estimated
## [4,] alpha     normal       Estimated
##   Variance-covariance matrix:
##       b0 b1 h0 alpha
## b0     1  0  0     0
## b1     0  1  0     0
## h0     0  0  0     0
## alpha  0  0  0     0
##   Error model: constant , initial values: a.1=1
##     No covariate in the model.
##     Initial values
##              b0   b1  h0 alpha
## Pop.CondInit 73 1.25 0.6 1e-04
## ----------------------------------
## ----    Key algorithm options  ----
## ----------------------------------
##     Estimation of individual parameters (MAP)
##     Estimation of standard errors and linearised log-likelihood
```

```
##     Estimation of log-likelihood by importance sampling
##     Number of iterations:  K1=300, K2=100
##     Number of chains:  1
##     Seed:  12345
##     Number of MCMC iterations for IS:  5000
##     Simulations:
##         nb of simulated datasets used for npde:  1000
##         nb of simulated datasets used for VPC:  100
##     Input/output
##         save the results to a file:  TRUE
##         save the graphs to files:  FALSE
##         directory where results should be saved:  newdir
## ----------------------------------------------------
## ----                  Results                   ----
## ----------------------------------------------------
## -----------------  Fixed effects  ------------------
## ----------------------------------------------------
##      Parameter Estimate
## [1,] b0        73.338
## [2,] b1         0.570
## [3,] h0         3.094
## [4,] alpha     -0.039
## [5,] a.1       17.233
## ----------------------------------------------------
## -----------  Variance of random effects  -----------
## ----------------------------------------------------
##    Parameter Estimate
## b0 omega2.b0 369
## b1 omega2.b1  16
## ----------------------------------------------------
## ------  Correlation matrix of random effects  ------
## ----------------------------------------------------
##           omega2.b0 omega2.b1
## omega2.b0 1         0
## omega2.b1 0         1
## ----------------------------------------------------
## ---------------  Statistical criteria  -------------
## ----------------------------------------------------
##
## Likelihood computed by importance sampling
##       -2LL= 28050.73
##       AIC = 28064.73 28064.73
##       BIC = 28094.07 28094.07
## ----------------------------------------------------
```

We obtain the summary of the fit with the parameter estimates, and likelihood value at MLE (with AIC and BIC). The standard error estimates are obtained using the following script.

```
yfit@results@fim  # variance covariance matrix (inverse of the FIM)
```

```
##                 b0           b1            h0         alpha     omega2.b0
## [1,]  1.0004032819 -5.975768e-02  0.0232522122 -3.188424e-04   2.569130851
## [2,] -0.0597576780  1.087968e-01 -0.0035186342  7.009299e-05  -0.088465304
```

```
## [3,]  0.0232522122 -3.518634e-03  0.0441352091 -6.058584e-04  -0.371007774
## [4,] -0.0003188424  7.009299e-05 -0.0006058584  9.059682e-06   0.002326825
## [5,]  2.5691308512 -8.846530e-02 -0.3710077744  2.326825e-03 988.446907983
## [6,]  0.0110253172  1.692256e-01 -0.0105168727  4.847003e-05   2.419921193
## [7,] -0.0280793906 -4.166982e-03  0.0005563514 -2.434968e-05  -0.147787916
##          omega2.b1           a.1
## [1,]  1.102532e-02 -2.807939e-02
## [2,]  1.692256e-01 -4.166982e-03
## [3,] -1.051687e-02  5.563514e-04
## [4,]  4.847003e-05 -2.434968e-05
## [5,]  2.419921e+00 -1.477879e-01
## [6,]  3.970566e+00 -7.040594e-02
## [7,] -7.040594e-02  2.981143e-02
```

```
sqrt(diag(yfit@results@fim)) # standard errors of parameters estimates
```

```
## [1]  1.00020162  0.32984363  0.21008381  0.00300993 31.43957551  1.99262797
## [7]  0.17265985
```

```
# Formatting results in a data frame
d = data.frame(par=c(yfit@results@name.fixed,yfit@results@name.random,"sigma_a1"),
               est = c(yfit@results@fixed.effects,diag(yfit@results@omega)[1:2],
                       yfit@results@respar[c(1)]),
               se = sqrt(diag(yfit@results@fim)))
print(d)
```

```
##         par          est          se
## 1        b0  73.33846825  1.00020162
## 2        b1   0.57019224  0.32984363
## 3        h0   3.09438337  0.21008381
## 4     alpha  -0.03867724  0.00300993
## 5 omega2.b0 369.32691471 31.43957551
## 6 omega2.b1  15.75845002  1.99262797
## 7  sigma_a1  17.23307846  0.17265985
```

COVID-19 example (multivariate joint model with competing risks)

We consider the joint model defined in the article involving three biomarkers and two competing risks:

$$
\begin{aligned}
y_{ij1} &= m_1(t_{ij1}, \psi_{i1}) + \sigma_1\, m_1(t_{ij1}, \psi_{i1})\epsilon_{ij1} \\
y_{ij2} &= m_2(t_{ij2}, \psi_{i2}) + \sigma_2\epsilon_{ij2} \\
y_{ij3} &= m_3(t_{ij3}, \psi_{i3}) + \sigma_3\epsilon_{ij3}
\end{aligned}
\tag{2}
$$

$$
h_{1i}(t, \psi_i, Score_i; \theta) = h_0 \exp\left[\sum_{k=1}^{3}\Big(\alpha_{1k} \times \big(m_k(t, \psi_{ik}) - med_k\big)\Big) + \beta \cdot Score_i\right]
$$

$$
h_{2i}(t, \psi_i, Score_i; \theta) = \frac{1}{b}\frac{(1 - F_1(\infty))\exp(-t/b)}{1 - (1 - F_1(\infty))\,(1 - \exp(-t/b))}
$$

with:

$$
\begin{aligned}
m_1(t_{ij1}, \psi_{i1}) &= \psi_{i01} + \psi_{ia1} \times [\exp(\psi_{i11}\, t_{ij1}) - \exp(\psi_{i21}\, t_{ij1})] \\
m_2(t_{ij2}, \psi_{i2}) &= \psi_{i02} + \psi_{i12} \times t_{ij2} \\
m_3(t_{ij3}, \psi_{i3}) &= \psi_{i03} + \psi_{i13} \times t_{ij3}
\end{aligned} \tag{3}
$$

```
data_joint = read.table("W:/saemix/dev/riscov/dataJM.txt", header = T)
dataJM<-saemixData(name.data=data_joint, name.group=c("id"), name.predictors=c("time","obs"),
                   name.response="obs", name.ytype = "ytype", name.covariates = c("score4C"))
```

Creating the saemix data object

```
##
##
## The following SaemixData object was successfully created:
##
## Object of class SaemixData
##     longitudinal data for use with the SAEM algorithm
## Dataset data_joint
##     Structured data: obs ~ time + obs | id
##     X variable for graphs: time ()
##     covariates: score4C (-)
```

Creating the saemix model object The model can be implemented as follows.

```
JMmodel<-function(psi,id,xidep) {
    ytype<-xidep$ytype
    N = unique(id)

    b01 <- psi[id,1]
    b11 <- psi[id,2]
    b21 <- psi[id,3]
    a1 <- psi[id,4]
    b02 <- psi[id,5]
    b12 <- psi[id,6]
    b03 <- psi[id,7]
    b13 <- psi[id,8]
    h1 <- psi[id,9]
    alpha1 <- psi[id,10]
    alpha2 <- psi[id,11]
    alpha3 <- psi[id,12]
    b <- psi[id,13]
    cov <- psi[id,14]

    T2 = xidep[ytype==2,1]  # vector of times for biom 2
    T3 = xidep[ytype==3,1]  # vector of times for biom 3
    T<-xidep[ytype==4,1]    # vector of times partie survie
    ev = xidep$obs[ytype==4]
    Nj <- length(T)
    cens<-which(ev==0)  # index of censored observations
```

```r
ind1 <- which(ev==1) # index of event 1
ind2 <- which(ev==2) # index of event 2

schem = sapply(N, function(i) length(which(id <= i)))
# schem is a vector giving the number of measurements for each patients
# defined for having unique individual parameters (see what follows)

b01b <- b01[schem]
b11b <- b11[schem]
b21b = b21[schem]
a1b = a1[schem]
b02b <- b02[schem]
b12b <- b12[schem]
b03b = b03[schem]
b13b = b13[schem]
h1b <- h1[schem]
alpha1b = alpha1[schem]
alpha2b = alpha2[schem]
alpha3b = alpha3[schem]
bb = b[schem]
covb = cov[schem]

f=function(x) seq(0,x,length.out=100)
tab = mapply(f,T)
tab = t(tab)
pas = tab[,2]-tab[,1]
# used for numerical integration because no explicit form of the cumulative hazard

f2=function(x) seq(0,x,length.out=1000)
tab2 = replicate(Nj,f2(1000))
tab2 = t(tab2)
pas2 = tab2[,2]-tab2[,1]
# used to compute the value F1(inf) (used in the competing event hazard)
# proxy: F1(inf) approximated by F1(1000)

haz1 = h1b*exp(alpha1b*(b01b+a1b*(exp(b11b*tab)-exp(b21b*tab))-5.78)
               +alpha2b*(b02b+b12b*tab-7.42)
               +alpha3b*(b03b+b13b*tab-2.89)+covb)
H1 = apply(haz1,1,sum)*pas # cumulative hazard from 0 to T
hazt1 = haz1[,100] # cumulative hazard for event 1 at time T

haz1b = h1b*exp(alpha1b*(b01b+a1b*(exp(b11b*tab2)-exp(b21b*tab2))-5.78)
                +alpha2b*(b02b+b12b*tab2-7.42)
                +alpha3b*(b03b+b13b*tab2-2.89)+covb)
H1b = apply(haz1b,1,sum)*pas2
F1 = 1-exp(-H1b) # F1(1000)

hazt2 = 1/bb*(1-F1)*exp(-T/bb)/(1-(1-F1)*(1-exp(-T/bb)))
H2 = -log(1-(1-F1)*(1-exp(-T/bb))) # cumulative hazard for event 2 at time T

logpdf <- rep(0,Nj)
logpdf[cens] <- log(exp(-H1[cens])+exp(-H2[cens])-1)
# likelihood contributions for censored observations
```

```
    logpdf[ind1] <- -H1[ind1] + log(hazt1[ind1])
    # likelihood contributions for event 1 observations
    logpdf[ind2] <- -H2[ind2] + log(hazt2[ind2])
    # likelihood contributions for event 2 observations

    ypred = rep(NA,length(xidep[,1]))

    ypred[ytype==1] = b01[ytype==1]+a1[ytype==1]*(exp(b11[ytype==1]*xidep[ytype==1,1])
                                                  -exp(b21[ytype==1]*xidep[ytype==1,1]))
    ypred[ytype==2] = b02[ytype==2]+b12[ytype==2]*xidep[ytype==2,1]
    ypred[ytype==3] = b03[ytype==3]+b13[ytype==3]*xidep[ytype==3,1]

    ypred[ytype==4] = logpdf

    return(ypred)
}

# Initial guesses for parameter estimates
param_longi<-c(4.5,-0.14,-0.16,6.1,7.4,0.0008,4.2,-0.15)
omega_longi <- c(4,0.01,0.003,0.7,0.001,0.00001,0.8,0.01)
param_surv<-c(0.0009,0.25,-9,0.76,16,0)
omega_surv <- c(0.01,0.01,0.01,0.01,0.01,0.01)
param_cov <-  c(0,0,0,0,0,0,0,0,0,0,0,0,0,0.3)
param <- matrix(data = c(param_longi, param_surv, param_cov), ncol = 14, byrow = T,
                dimnames=list(NULL, c( "b01", "b11", "b21", "a1","b02", "b12", "b03", "b13",
                                       "h1", "alpha1", "alpha2", "alpha3", "b", "cov")))
omega <- c(omega_longi, omega_surv)

jointTTE<-saemixModel(model=JMmodel,description="JM lin+competing risks",
                      modeltype=c("structural","structural","structural","likelihood"),
                      psi0=param,transform.par=c(0,0,0,1,0,0,0,0,1,0,0,0,1,0),
                      covariance.model=diag(c(1,1,1,1,1,1,1,1,0,0,0,0,0,0)),
                      omega.init = diag(omega), fixed.estim = c(1,1,1,1,1,1,1,1,1,1,1,1,1,0),
                      covariate.model = c(0,0,0,0,0,0,0,0,0,0,0,0,0,1),
                      error.model = c("proportional","constant","constant","likelihood"))
```

```
##
##
## The following SaemixModel object was successfully created:
##
## Nonlinear mixed-effects model
##   Model function:  JM lin+competing risks
##   Model type:  structural structural structural likelihood
## function(psi,id,xidep) {
##     ytype<-xidep$ytype
##     N = unique(id)
##
##     b01 <- psi[id,1]
##     b11 <- psi[id,2]
##     b21 <- psi[id,3]
##     a1 <- psi[id,4]
##     b02 <- psi[id,5]
##     b12 <- psi[id,6]
```

```
##     b03 <- psi[id,7]
##     b13 <- psi[id,8]
##     h1 <- psi[id,9]
##     alpha1 <- psi[id,10]
##     alpha2 <- psi[id,11]
##     alpha3 <- psi[id,12]
##     b <- psi[id,13]
##     cov <- psi[id,14]
##
##     T2 = xidep[ytype==2,1]  # vector of times for biom 2
##     T3 = xidep[ytype==3,1]  # vector of times for biom 3
##     T<-xidep[ytype==4,1]    # vector of times partie survie
##     ev = xidep$obs[ytype==4]
##     Nj <- length(T)
##     cens<-which(ev==0)  # index of censored observations
##     ind1 <- which(ev==1) # index of event 1
##     ind2 <- which(ev==2) # index of event 2
##
##     schem = sapply(N, function(i) length(which(id <= i)))
##     # schem is a vector giving the number of measurements for each patients
##     # defined for having unique individual parameters (see what follows)
##
##     b01b <- b01[schem]
##     b11b <- b11[schem]
##     b21b = b21[schem]
##     a1b = a1[schem]
##     b02b <- b02[schem]
##     b12b <- b12[schem]
##     b03b = b03[schem]
##     b13b = b13[schem]
##     h1b <- h1[schem]
##     alpha1b = alpha1[schem]
##     alpha2b = alpha2[schem]
##     alpha3b = alpha3[schem]
##     bb = b[schem]
##     covb = cov[schem]
##
##     f=function(x) seq(0,x,length.out=100)
##     tab = mapply(f,T)
##     tab = t(tab)
##     pas = tab[,2]-tab[,1]
##     # used for numerical integration because no explicit form of the cumulative hazard
##
##     f2=function(x) seq(0,x,length.out=1000)
##     tab2 = replicate(Nj,f2(1000))
##     tab2 = t(tab2)
##     pas2 = tab2[,2]-tab2[,1]
##     # used to compute the value F1(inf) (used in the competing event hazard)
##     # proxy: F1(inf) approximated by F1(1000)
##
##     haz1 = h1b*exp(alpha1b*(b01b+a1b*(exp(b11b*tab)-exp(b21b*tab))-5.78)
##                    +alpha2b*(b02b+b12b*tab-7.42)
##                    +alpha3b*(b03b+b13b*tab-2.89)+covb)
##     H1 = apply(haz1,1,sum)*pas # cumulative hazard from 0 to T
```

```
##     hazt1 = haz1[,100] # cumulative hazard for event 1 at time T
##
##     haz1b = h1b*exp(alpha1b*(b01b+a1b*(exp(b11b*tab2)-exp(b21b*tab2))-5.78)
##                     +alpha2b*(b02b+b12b*tab2-7.42)
##                     +alpha3b*(b03b+b13b*tab2-2.89)+covb)
##     H1b = apply(haz1b,1,sum)*pas2
##     F1 = 1-exp(-H1b) # F1(1000)
##
##     hazt2 = 1/bb*(1-F1)*exp(-T/bb)/(1-(1-F1)*(1-exp(-T/bb)))
##     H2 = -log(1-(1-F1)*(1-exp(-T/bb))) # cumulative hazard for event 2 at time T
##
##     logpdf <- rep(0,Nj)
##     logpdf[cens] <- log(exp(-H1[cens])+exp(-H2[cens])-1)
##     # likelihood contributions for censored observations
##     logpdf[ind1] <- -H1[ind1] + log(hazt1[ind1])
##     # likelihood contributions for event 1 observations
##     logpdf[ind2] <- -H2[ind2] + log(hazt2[ind2])
##     # likelihood contributions for event 2 observations
##
##     ypred = rep(NA,length(xidep[,1]))
##
##     ypred[ytype==1] = b01[ytype==1]+a1[ytype==1]*(exp(b11[ytype==1]*xidep[ytype==1,1])
##                                                   -exp(b21[ytype==1]*xidep[ytype==1,1]))
##     ypred[ytype==2] = b02[ytype==2]+b12[ytype==2]*xidep[ytype==2,1]
##     ypred[ytype==3] = b03[ytype==3]+b13[ytype==3]*xidep[ytype==3,1]
##
##     ypred[ytype==4] = logpdf
##
##     return(ypred)
## }
##   Nb of parameters: 14
##       parameter names:  b01 b11 b21 a1 b02 b12 b03 b13 h1 alpha1 alpha2 alpha3 b cov
##       distribution:
##       Parameter Distribution Estimated
##  [1,] b01       normal       Estimated
##  [2,] b11       normal       Estimated
##  [3,] b21       normal       Estimated
##  [4,] a1        log-normal   Estimated
##  [5,] b02       normal       Estimated
##  [6,] b12       normal       Estimated
##  [7,] b03       normal       Estimated
##  [8,] b13       normal       Estimated
##  [9,] h1        log-normal   Estimated
## [10,] alpha1    normal       Estimated
## [11,] alpha2    normal       Estimated
## [12,] alpha3    normal       Estimated
## [13,] b         log-normal   Estimated
## [14,] cov       normal       Fixed
##   Variance-covariance matrix:
##        b01 b11 b21 a1 b02 b12 b03 b13 h1 alpha1 alpha2 alpha3 b cov
## b01      1   0   0  0   0   0   0   0  0      0      0      0 0   0
## b11      0   1   0  0   0   0   0   0  0      0      0      0 0   0
## b21      0   0   1  0   0   0   0   0  0      0      0      0 0   0
## a1       0   0   0  1   0   0   0   0  0      0      0      0 0   0
```

```
## b02      0   0   0  0   1   0   0   0  0      0      0      0 0   0
## b12      0   0   0  0   0   1   0   0  0      0      0      0 0   0
## b03      0   0   0  0   0   0   1   0  0      0      0      0 0   0
## b13      0   0   0  0   0   0   0   1  0      0      0      0 0   0
## h1       0   0   0  0   0   0   0   0  0      0      0      0 0   0
## alpha1   0   0   0  0   0   0   0   0  0      0      0      0 0   0
## alpha2   0   0   0  0   0   0   0   0  0      0      0      0 0   0
## alpha3   0   0   0  0   0   0   0   0  0      0      0      0 0   0
## b        0   0   0  0   0   0   0   0  0      0      0      0 0   0
## cov      0   0   0  0   0   0   0   0  0      0      0      0 0   0
##   Error model: proportional , initial values: b.1=1 a.1=1 a.1=1
##   Error model: proportional , initial values: b.1=1 a.1=1 a.1=1
##   Error model: proportional , initial values: b.1=1 a.1=1 a.1=1
##   Covariate model:
##      b01 b11 b21 a1 b02 b12 b03 b13 h1 alpha1 alpha2 alpha3 b cov
## [1,]   0   0   0  0   0   0   0   0  0      0      0      0 0   1
##     Initial values
##              b01   b11   b21  a1 b02   b12 b03   b13    h1 alpha1 alpha2 alpha3
## Pop.CondInit 4.5 -0.14 -0.16 6.1 7.4 8e-04 4.2 -0.15 9e-04   0.25     -9   0.76
## Cov.CondInit 0.0  0.00  0.00 0.0 0.0 0e+00 0.0  0.00 0e+00   0.00      0   0.00
##               b cov
## Pop.CondInit 16 0.0
## Cov.CondInit  0 0.3
```