Saemix 3 - time-to-event data models

Emmanuelle

09/2022

Version

Use saemix version ≥ 3.2

Objective

Run TTE and RTTE models in **saemix**

This notebook uses additional result files from the **saemix** development github (https://github.com/saemixdevelopment/saemixextension), not integrated in the package to avoid bloating. The *workDir* folder in the next chunk of code points to the folder where the user stored this code, and is needed to run the notebook (*workDir* defaults to the current working directory). Specifically, the notebook loads the results for the bootstrap runs performed using different approaches (see Comets et al. Pharm Res 2021). Bootstraps can be run instead by switching the *runBootstrap* variable to TRUE in the first chunk of code:

- in the code, the number of bootstraps is set to 10 for speed but we recommend to use at least 200 for a 90% CI.
- this can be changed in the following change of code by uncommenting the line *nboot<-200* and setting the number of bootstrap samples (this may cause memory issues in **Rstudio** with older machines, if this is the case we recommend executing the code in a separate script)

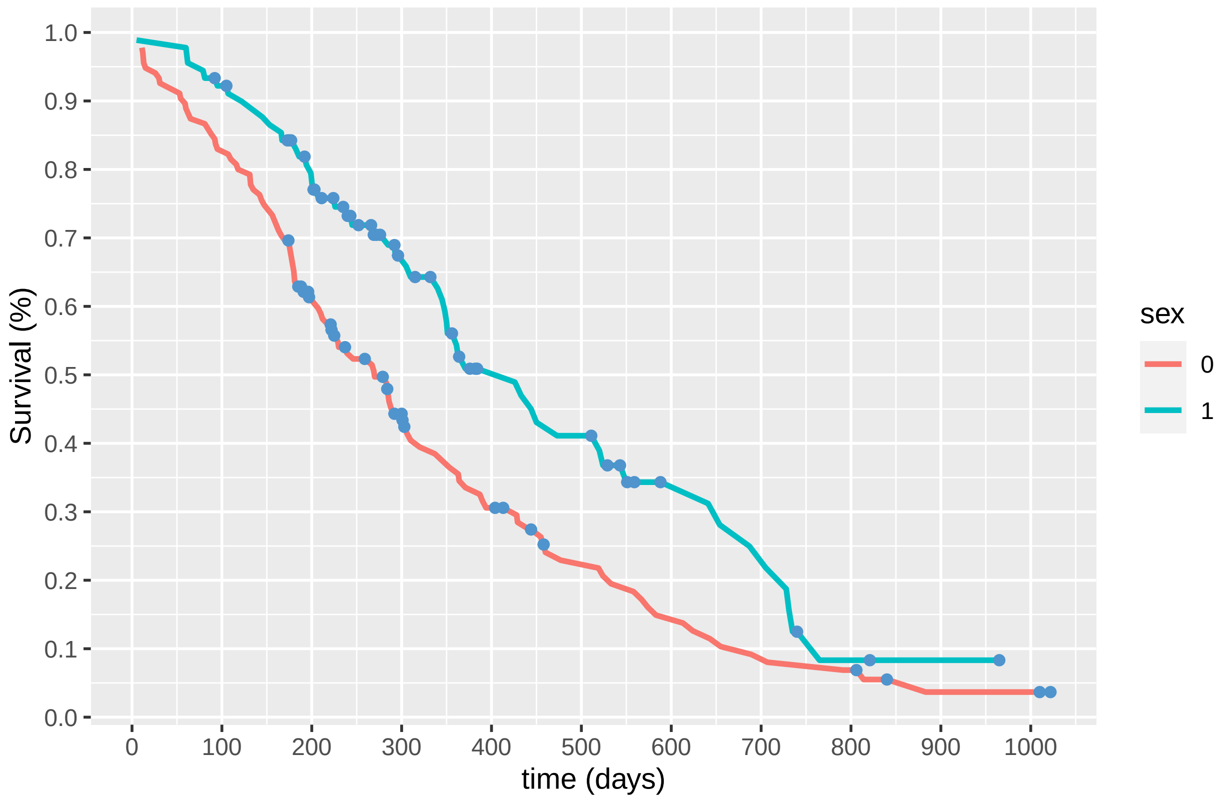
The current notebook can be executed to create an HMTL or PDF output with comments and explanations. A script version containing only the R code is also given as *saemix3_tteModel.R* in the same folder.

TTE data

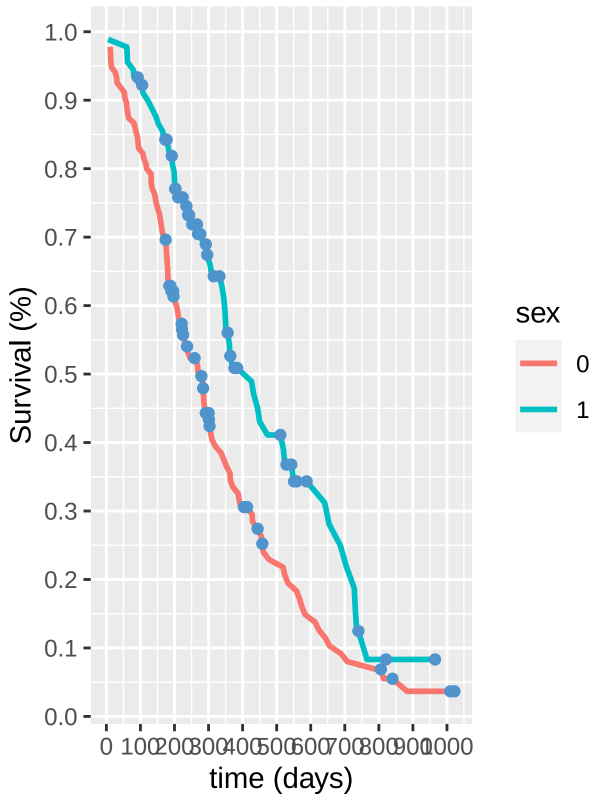
Data description - lung cancer The example chosen to illustrate the analysis of time-to-event data in **saemix** is the NCCTG Lung Cancer Data, describing the survival in patients with advanced lung cancer from the North Central Cancer Treatment Group (Loprinzi et al. 1994). Covariates measured in the study include performance scores rating how well the patient can perform usual daily activities. We reformatted the *cancer* dataset (previously *lung*) provided in the **survival** package in R in SAEM format: patients with missing age, sex, institution or physician assessments were removed from the dataset (3 patients removed, leaving 225 in the database). Status was recoded as 1 for death and 0 for a censored event, and a censoring column was added to denote whether the patient was dead or alive at the time of the last observation. A line at time=0 was added for all subjects.

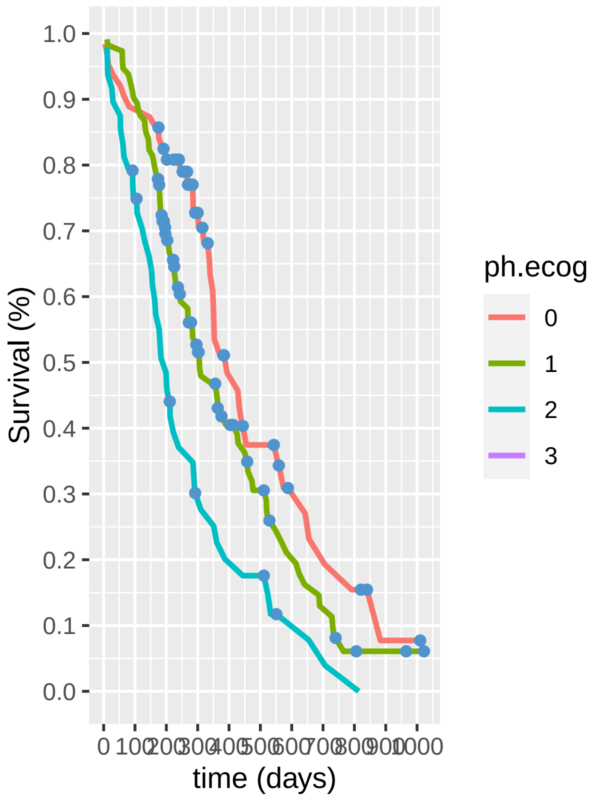
We can plot the distribution of times as a histogram. Note that this dataset contains many missing values in the meal calorie column, and a few in weight loss and patient-assessed Karnofsky score. Here we set the missing patient-assessed Karnofsky scores to the median, and we don't include the other two covariates in the dataset as they have more missing values.

```r
data(lung.saemix)
# all covariates (but need to manage the missing covariates)
# ECOG status treated as continuous
# missing patient Karnofsky scores set to median (in 3 patients)
# other covariates still have missing values
lung1<-lung.saemix
lung1$pat.karno[is.na(lung1$pat.karno)]<-median(lung1$pat.karno, na.rm=TRUE)

saemix.data.contPH<-saemixData(name.data=lung1,header=TRUE,name.group=c("id"),
                    name.predictors=c("time","status","cens"),name.response=c("status"),
                    name.covariates=c( "sex", "ph.ecog", "ph.karno", "pat.karno", "age"),
                    units=list(x="days",y="",covariates=c("","-","%","%","yr")), verbose=FALSE)

plotDiscreteData(saemix.data.contPH, outcome="tte", which.cov="sex")
```

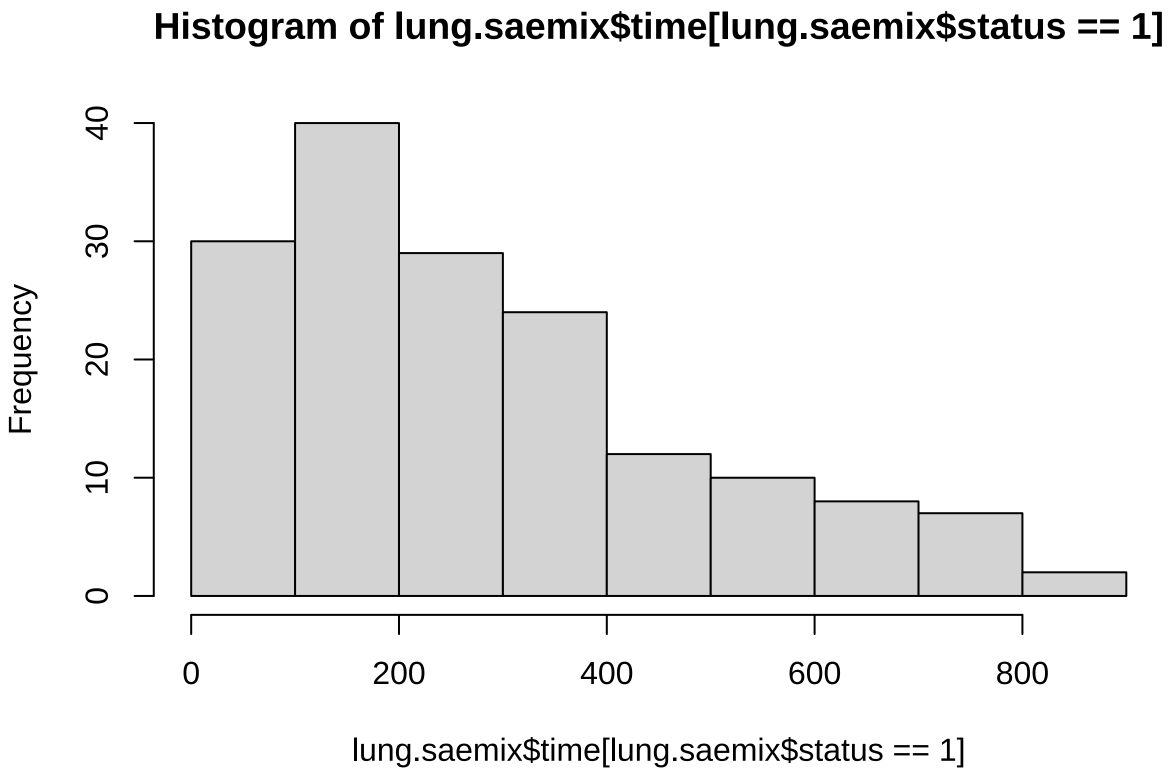


```
xplot1<-plotDiscreteData(saemix.data.contPH, outcome="tte", which.cov="sex")
xplot2<-plotDiscreteData(saemix.data.contPH, outcome="tte", which.cov="ph.ecog")
grid.arrange(grobs=list(xplot1, xplot2), nrow=1, ncol=2)
```





```
# Histogram
hist(lung.saemix$time[lung.saemix$status==1])
```



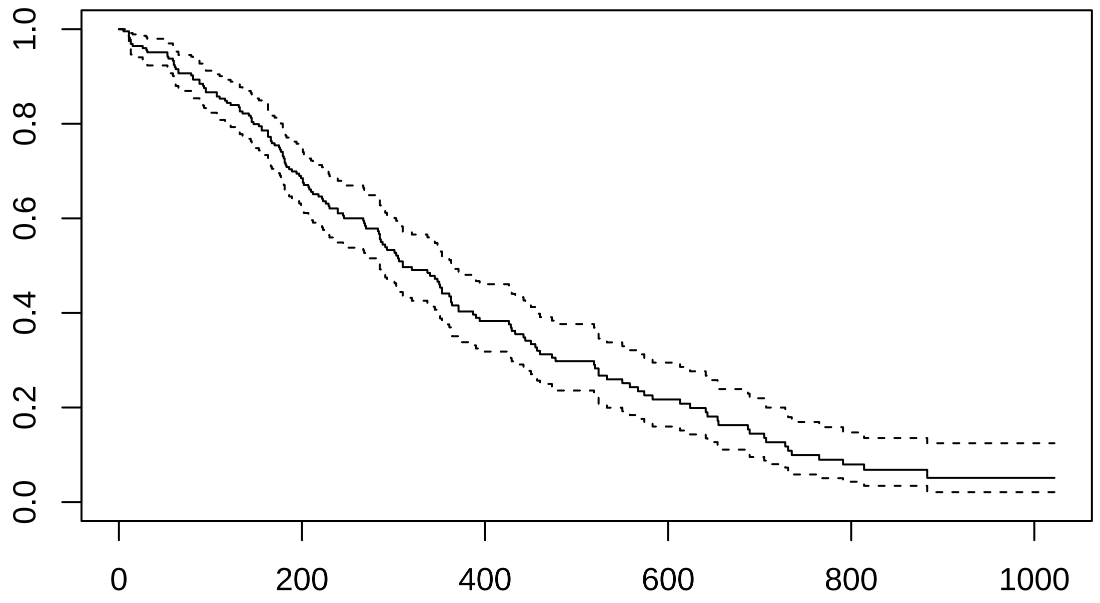
```
# Note: missing data in pat.karno, wt.loss and meal.cal
if(FALSE)
    print(summary(lung.saemix))

if(saveFigs) {
  namfig<-"lung_exploreSurv.eps"
  cairo_ps(file = file.path(figDir, namfig), onefile = TRUE, fallback_resolution = 600, height=8.27, wi
  grid.arrange(grobs=list(xplot1, xplot2), nrow=1, ncol=2)
  dev.off()
}
```

Kaplan-Meier plot We can also plot the traditional Kaplan-Meier plot using the functions in the **survival** package.

```
lung.surv<-lung.saemix[lung.saemix$time>0,]
lung.surv$status<-lung.surv$status+1
Surv(lung.surv$time, lung.surv$status) # 1=censored, 2=dead
```

```
##   [1]  306   455   1010+  210   883   1022+  310   361   218   166   170   654
##  [13]  728   567   144   613   707    61    88   301    81   624   371   394
##  [25]  520   574   118   390    12   473    26   533   107    53   122   814
##  [37]  965+   93   731   460   153   433   145   583    95   303   519   643
##  [49]  765   735   189    53   246   689    65     5   132   687   345   444
##  [61]  223   175    60   163    65   208   821+  428   230   840+  305    11
##  [73]  132   226   426   705   363    11   176   791    95   196+  167   806+
##  [85]  284   641   147   740+  163   655   239    88   245   588+   30   179
##  [97]  310   477   166   559+  450   364   107   177   156   529+   11   429
## [109]  351    15   181   283   201   524    13   212   524   288   363   442
## [121]  199   550    54   558   207    92    60   551+  543+  293   202   353
## [133]  511+  267   511+  371   387   457   337   201   404+  222    62   458+
## [145]  356+  353   163    31   340   229   444+  315+  182   156   364+  291
## [157]  179   376+  384+  268   292+  142   413+  266+  194   320   181   285
## [169]  301+  348   197   382+  303+  296+  180   186   145   269+  300+  284+
## [181]  350   272+  292+  332+  285   259+  110   286   270    81   131   225+
## [193]  269   225+  243+  279+  276+  135    79    59   240+  202+  235+  224+
## [205]  239   237+  173+  252+  221+  185+   92+   13   222+  192+  183   211+
## [217]  175+  197+  203+  116   188+  191+  105+  174+  177+
```

```
nonpar.fit <- survfit(Surv(time, status) ~ 1, data = lung.surv)
plot(nonpar.fit)
```



Model for TTE data In **saemix**, we model survival using parametric models. Here we can first use a Weibull model for the hazard, parameterised as T_e and γ. For individual i, the hazard function of this model is:

$$h(t) = \frac{\gamma}{T_e} \left(\frac{t}{T_e} \right)^{\gamma-1}$$

And the parametric survival function is given by:

$$S(t) = e^{-\left(\frac{t}{T_e}\right)^{\gamma}} = e^{-H(t)}$$

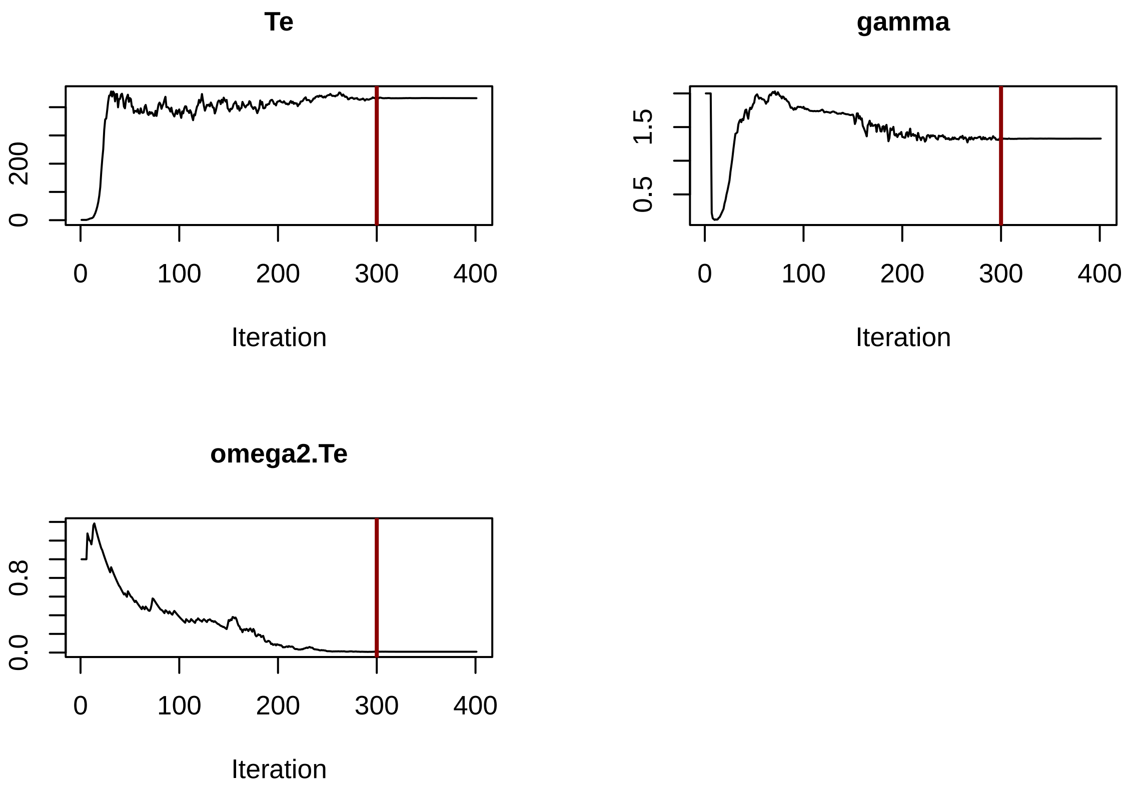
where H denotes the cumulative hazard function.

In the model function in **saemix**, we define the log-likelihood of each event in the dataset:

- at time 0, we set a log-likelihood of 0
- at the time of an event, the likelihood is equal to $l(t, \delta, \psi_i$ where δ is the censoring indicator (1 if the event occurred, 0 if the time corresponds to a censoring time) and $\psi_i = (T_{e,i}, \gamma_i)$ are the individual parameters for subject i.

```
weibulltte.model<-function(psi,id,xidep) {
  T<-xidep[,1]
  y<-xidep[,2] # events (1=event, 0=no event)
  cens<-which(xidep[,3]==1) # censoring times (subject specific)
  init <- which(T==0)
  Te <- psi[id,1] # Parameters of the Weibull model
  gamma <- psi[id,2]
  Nj <- length(T)

  ind <- setdiff(1:Nj, append(init,cens)) # indices of events
  hazard <- (gamma/Te)*(T/Te)^(gamma-1) # h
  H <- (T/Te)^gamma # H= -ln(S)
  logpdf <- rep(0,Nj) # ln(l(T=0))=0
  logpdf[cens] <- -H[cens] + H[cens-1] # ln(l(T=censoring time))=ln(S)=-H
  logpdf[ind] <- -H[ind] + H[ind-1] + log(hazard[ind]) # ln(l(T=event time))=ln(S)+ln(h)
  return(logpdf)
}
```

```r
saemix.model<-saemixModel(model=weibulltte.model,description="Weibull TTE model",modeltype="likelihood"
  psi0=matrix(c(1,2),ncol=2,byrow=TRUE,dimnames=list(NULL,  c("Te","gamma"))),
  transform.par=c(1,1),covariance.model=matrix(c(1,0,0,0),ncol=2, byrow=TRUE), verbose=FALSE)
saemix.options<-list(seed=632545,save=FALSE,save.graphs=FALSE, displayProgress=FALSE, print=FALSE)
tte.fit<-saemix(saemix.model,saemix.data.contPH,saemix.options)
plot(tte.fit, plot.type="convergence")
```



```r
print(tte.fit)
```

```
## Nonlinear mixed-effects model fit by the SAEM algorithm
## -----------------------------------
## ----          Data             ----
## -----------------------------------
## Object of class SaemixData
##     longitudinal data for use with the SAEM algorithm
## Dataset lung1
##     Structured data: status ~ time + status + cens | id
##     X variable for graphs: time (days)
##     covariates: sex (), ph.ecog (-), ph.karno (%), pat.karno (%), age (yr)
##       reference class for covariate sex :  0
## Dataset characteristics:
##     number of subjects:     225
##     number of observations: 450
##     average/min/max nb obs: 2.00  /  2  /  2
## First 10 lines of data:
##    id time status cens status.1 sex ph.ecog ph.karno pat.karno age mdv cens.1
## 1   1    0      0    0        0   0       1       90       100  74   0      0
## 2   1  306      1    0        1   0       1       90       100  74   0      0
## 3   2    0      0    0        0   0       0       90        90  68   0      0
## 4   2  455      1    0        1   0       0       90         90  68   0      0
## 5   3    0      0    0        0   0       0       90         90  56   0      0
## 6   3 1010      0    1        0   0       0       90         90  56   0      0
## 7   4    0      0    0        0   0       1       90         60  57   0      0
## 8   4  210      1    0        1   0       1       90         60  57   0      0
## 9   5    0      0    0        0   0       0      100         90  60   0      0
## 10  5  883      1    0        1   0       0      100         90  60   0      0
##    occ ytype
## 1    1     1
## 2    1     1
## 3    1     1
## 4    1     1
## 5    1     1
## 6    1     1
## 7    1     1
## 8    1     1
## 9    1     1
## 10   1     1
## -----------------------------------
## ----          Model            ----
## -----------------------------------
## Nonlinear mixed-effects model
##   Model function:  Weibull TTE model
##   Model type:  likelihood
## function(psi,id,xidep) {
##   T<-xidep[,1]
##   y<-xidep[,2] # events (1=event, 0=no event)
##   cens<-which(xidep[,3]==1) # censoring times (subject specific)
##   init <- which(T==0)
##   Te <- psi[id,1] # Parameters of the Weibull model
##   gamma <- psi[id,2]
##   Nj <- length(T)
##
##   ind <- setdiff(1:Nj, append(init,cens)) # indices of events
##   hazard <- (gamma/Te)*(T/Te)^(gamma-1) # h
##   H <- (T/Te)^gamma # H= -ln(S)
##   logpdf <- rep(0,Nj) # ln(l(T=0))=0
##   logpdf[cens] <- -H[cens] + H[cens-1] # ln(l(T=censoring time))=ln(S)=-H
##   logpdf[ind] <- -H[ind] + H[ind-1] + log(hazard[ind]) # ln(l(T=event time))=ln(S)+ln(h)
##   return(logpdf)
## }
## <bytecode: 0x555e0c84fba8>
##   Nb of parameters: 2
##       parameter names:  Te gamma
##       distribution:
##      Parameter Distribution Estimated
## [1,] Te        log-normal   Estimated
## [2,] gamma     log-normal   Estimated
##   Variance-covariance matrix:
##       Te gamma
## Te     1     0
## gamma  0     0
##     No covariate in the model.
##     Initial values
##              Te gamma
## Pop.CondInit  1     2
## -----------------------------------
## ----    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:  632545
##     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:  FALSE
##         save the graphs to files:  FALSE
## ----------------------------------------------------
## ----                  Results                   ----
## ----------------------------------------------------
## -----------------  Fixed effects  ------------------
## ----------------------------------------------------
##      Parameter Estimate SE    CV(%)
## [1,] Te        431.8    51.60 12
## [2,] gamma       1.3     0.19 14
## ----------------------------------------------------
## -----------  Variance of random effects  -----------
## ----------------------------------------------------
##    Parameter Estimate SE   CV(%)
## Te omega2.Te 0.009    0.17 1858
## ----------------------------------------------------
## ------  Correlation matrix of random effects  ------
## ----------------------------------------------------
##           omega2.Te
## omega2.Te 1
## ----------------------------------------------------
## ---------------  Statistical criteria  -------------
## ----------------------------------------------------
## Likelihood computed by linearisation
##       -2LL= 5189.352
##       AIC = 5197.352
##       BIC = 5211.017
##
## Likelihood computed by importance sampling
##       -2LL= 2269.357
##       AIC = 2277.357
##       BIC = 2291.021
## ----------------------------------------------------
```

Simulation function Simulating from a TTE model is slightly more complicated than for the other non Gaussian models. When the hazard function has an inverse, we can use the inverse CDF technique (or inverse transformation algorithm) to generate random samples from the TTE model. The method uses the fact that a continuous cumulative density function, F, is a one-to-one mapping of the domain of the cdf into the interval

(0,1). Therefore, if U is a uniform random variable on (0,1), then $X = F^{˘1}(U)$ has the distribution F.

For the single event Weibull model:

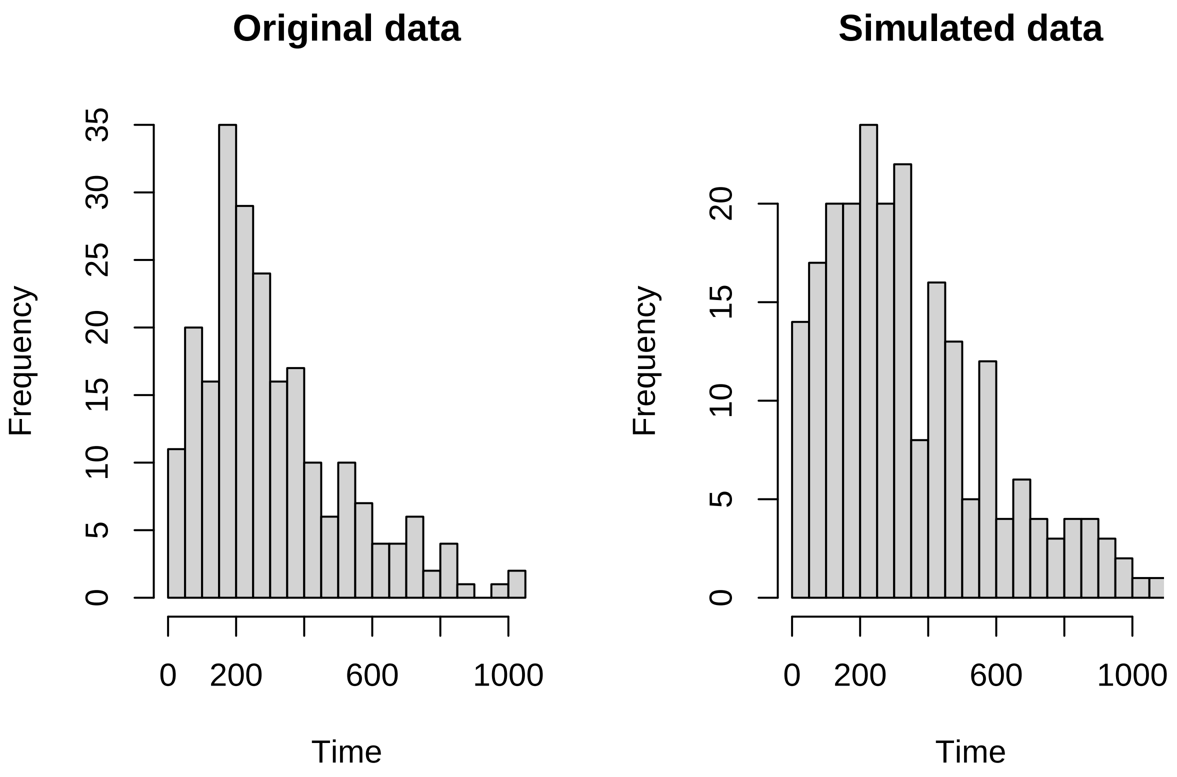
$$F = 1 - e^{-\int_0^T h(u)du} = 1 - e^{-\left(\frac{T}{T_e}\right)^\gamma} \sim \mathcal{U}(0,1)$$

Assuming we simulate $U = 1 - V$ from $\mathcal{U}(0,1)$, we can obtain a sample from the Weibull parametric model as:

$$T = T_e \left(-\ln(V)\right)^{1/\gamma}$$

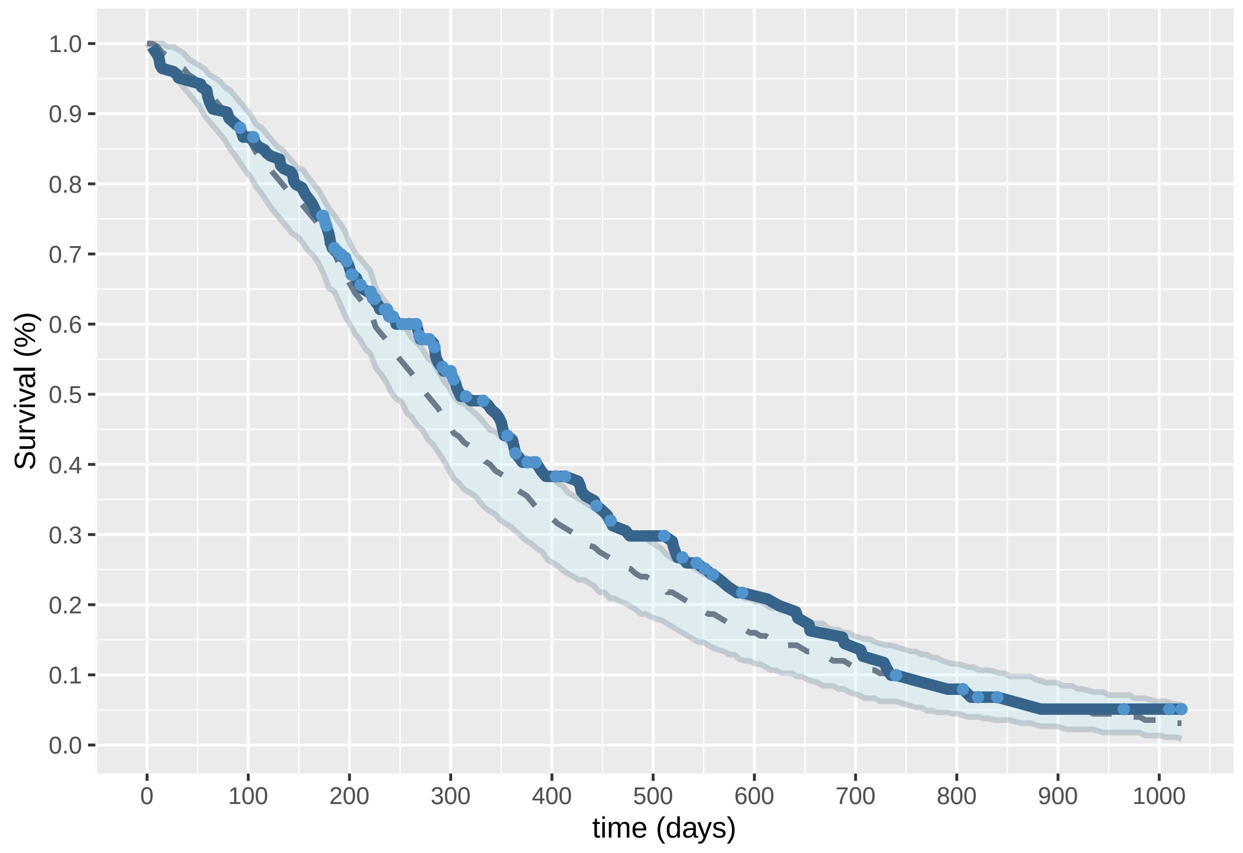
In the following we assume the first column of *xidep* contains the observed times, and we keep the censoring times recorded for each subject.

```
# Simulate events based on the observed individual censoring time
simulateWeibullTTE <- function(psi,id,xidep) {
  T<-xidep[,1]
  y<-xidep[,2] # events (1=event, 0=no event)
  delta <- xidep[,3] # censoring indicator
  cens<-which(xidep[,3]==1) # censoring times (subject specific)
  tmax <- max(T[cens]) # maximum censoring time observed in dataset
  init <- which(T==0)
  Te <- psi[,1] # Parameters of the Weibull model
  gamma <- psi[,2]
  Nj <- length(T)
  ind <- setdiff(1:Nj, append(init,cens)) # indices of events
  tevent<-T
  Vj<-runif(dim(psi)[1])
  tsim<-Te*(-log(Vj))^(1/gamma) #   events
  tevent[T>0]<-tsim
  tevent[delta==1 & tevent>T] <- T[delta==1 & tevent>T] # subject-specific censoring time
#  tevent[delta==0 & tevent>tmax] <- tmax # censoring to tmax (for subjects who experienced an event)
#  tevent[tevent[dead]>tmax] <- tmax # for subjects who initially experienced the event, use maximal ce
  return(tevent)
}

# Checking the simulation function
xidep1<-saemix.data.contPH@data[,saemix.data.contPH@name.predictors]
nsuj<-saemix.data.contPH@N
psiM<-data.frame(Te=rnorm(nsuj, mean=tte.fit@results@fixed.effects[1], sd=2), gamma=tte.fit@results@fixe
id1<-rep(1:nsuj, each=2)
simtime<-simulateWeibullTTE(psiM, id1, xidep1)

par(mfrow=c(1,2))
hist(saemix.data.contPH@data$time[saemix.data.contPH@data$time>0], breaks=30, xlim=c(0,1050),xlab="Time
hist(simtime[simtime>0], breaks=30, xlim=c(0,1050), xlab="Time", main="Simulated data")
```



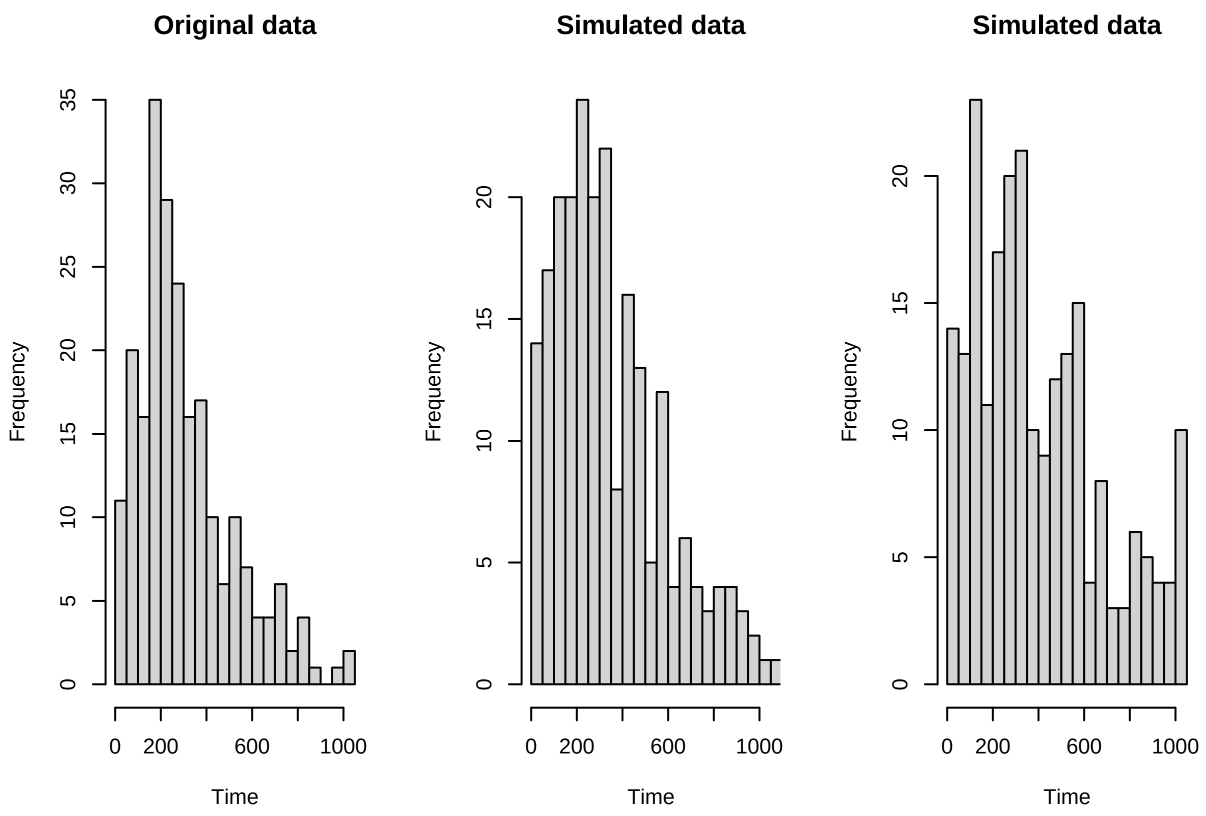
Diagnostics We add the simulation function to the model element of the fitted object (we can also include the simulation function when creating the model by adding the argument *simulate.function=simulateWeibullTTE* to saemixModel in the code above). We then simulate data using the fitted model and this function through the *simulateDiscreteSaemix()* function, and use the *discreteVPC()* function to obtain a Kaplan-Meier type VPC showing the prediction band for the survival function according to the model, overlaid with the actual observed survival function.

```
tte.fit@model@simulate.function <- simulateWeibullTTE
simtte.fit <- simulateDiscreteSaemix(tte.fit, nsim=500)

gpl <- discreteVPC(simtte.fit, outcome="TTE")
plot(gpl)
```



We could also assume a common censoring (function *simulateWeibullTTE.maxcens()* below) but simulating from this function shows an excess of times simulated at the censoring limit compared to the original dataset.

```
# Ignoring the cens column and assuming a common censoring time instead
simulateWeibullTTE.maxcens <- function(psi,id,xidep) {
  etime<-xidep[,1]
  censoringtime <- max(etime)
  Te <- psi[,1]
  gamma <- psi[,2]
  N<-dim(psi)[1]
  Vj<-runif(N)
  T<-Te*(-log(Vj))^(1/gamma)
  T[T>censoringtime]<-censoringtime
  etime[etime>0]<-T
  return(etime)
}
simtime.maxcens<-simulateWeibullTTE.maxcens(psiM, id1, xidep1)

par(mfrow=c(1,3))
hist(saemix.data.contPH@data$time[saemix.data.contPH@data$time>0], breaks=30, xlim=c(0,1050), xlab="Tim
hist(simtime[simtime>0], breaks=30, xlim=c(0,1050), xlab="Time", main="Simulated data")
hist(simtime.maxcens[simtime.maxcens>0], breaks=30, xlim=c(0,1050), xlab="Time", main="Simulated data")
```



Note that there are some specialised packages such as the **survsim** and the **simsurv** package that could be leveraged for this exercise. Also, a dedicated package was recently developed by Ron Keizer to implement VPC for different types of data. For survival data, we can also use the *vpc_tte()* function from this package to produce the KM-VPC plot (see additional script *saemix3_tteModel_ronVPC.R*).

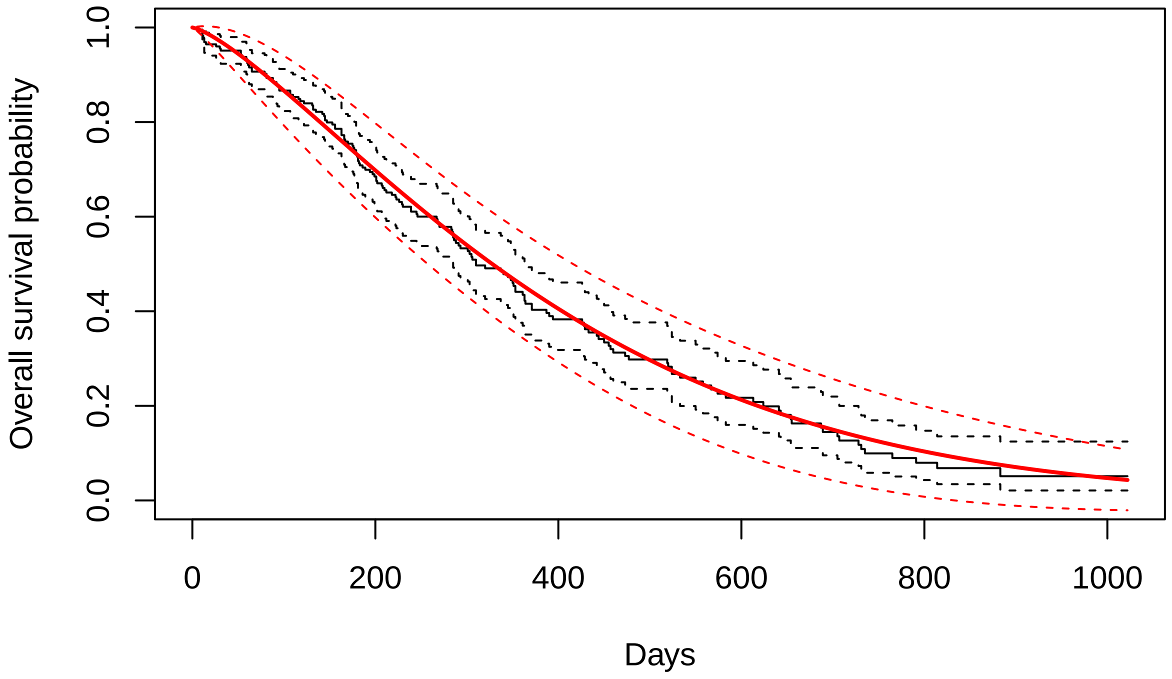
Comparison to the KM fit With TTE data the First-Order approximation for the FIM doesn't seem to perform too badly. We can use the delta-method to obtain standard errors around the value of the survival function, using the following vector of derivatives:

$$\begin{pmatrix} \frac{\delta S}{\delta T_e} \\ \frac{\delta S}{\delta \gamma} \end{pmatrix} = \begin{pmatrix} \frac{\gamma}{T_e} \left(\frac{t}{T_e}\right)^{\gamma} e^{-\left(\frac{t}{T_e}\right)^{\gamma}} \\ -\ln\left(\frac{t}{T_e}\right) \left(\frac{t}{T_e}\right)^{\gamma} e^{-\left(\frac{t}{T_e}\right)^{\gamma}} \end{pmatrix}$$

We overlay the parametric fit and its confidence interval in red over the previous non-parametric KM estimate, and find a good concordance between the two.

```
ypred<-predict(tte.fit)

# Use survival package to assess Survival curve
xtim<-seq(0,max(lung.saemix$time), length.out=200)
estpar<-tte.fit@results@fixed.effects
estse<-tte.fit@results@se.fixed
ypred<-exp(-(xtim/estpar[1])^(estpar[2]))

# Computing SE for the survival curve based on linearised FIM (probably not a good idea) through the de
invfim<-solve(tte.fit@results@fim[1:2,1:2])
xcal<- (xtim/estpar[1])^estpar[2]
dsdgamma<- -log(xtim/estpar[1]) * xcal *exp(-xcal)
dsdalpha<- estpar[2]/estpar[1] * xcal *exp(-xcal)
```

```r
xmat<-rbind(dsdalpha, dsdgamma)
#    x1<-t(xmat[,1:3]) %*% invfim %*% xmat[,1:3]
sesurv<-rep(0,length(xcal))
for(i in 1:length(xcal))
  sesurv[i]<-sqrt(t(xmat[,i]) %*% invfim %*% xmat[,i])

# Comparison between KM and parametric fit
plot(nonpar.fit, xlab = "Days", ylab = "Overall survival probability")
lines(xtim,ypred, col="red",lwd=2)
lines(xtim,ypred+1.96*sesurv, col="red",lwd=1, lty=2)
lines(xtim,ypred-1.96*sesurv, col="red",lwd=1, lty=2)
```



Selecting a parametric model We now consider alternative models to fit the same data. Given the shape of the survival functions, other classical models we can consider are the exponential (or constant hazard) model, the log-logistic model, the gamma model and the Gompertz model. The corresponding hazard functions are: - exponential model:

$$h(t) = \frac{1}{T_e}$$

- Gompertz:

$$h(t) = \frac{\gamma}{T'_e} \, e^{\frac{t}{T'_e}} \gamma \, (e^{\frac{t}{T'_e}} - 1)$$

where

$$T'_e = \frac{T_e}{ln\left(1 + \frac{ln(2)}{\gamma}\right)}$$

- gamma model:

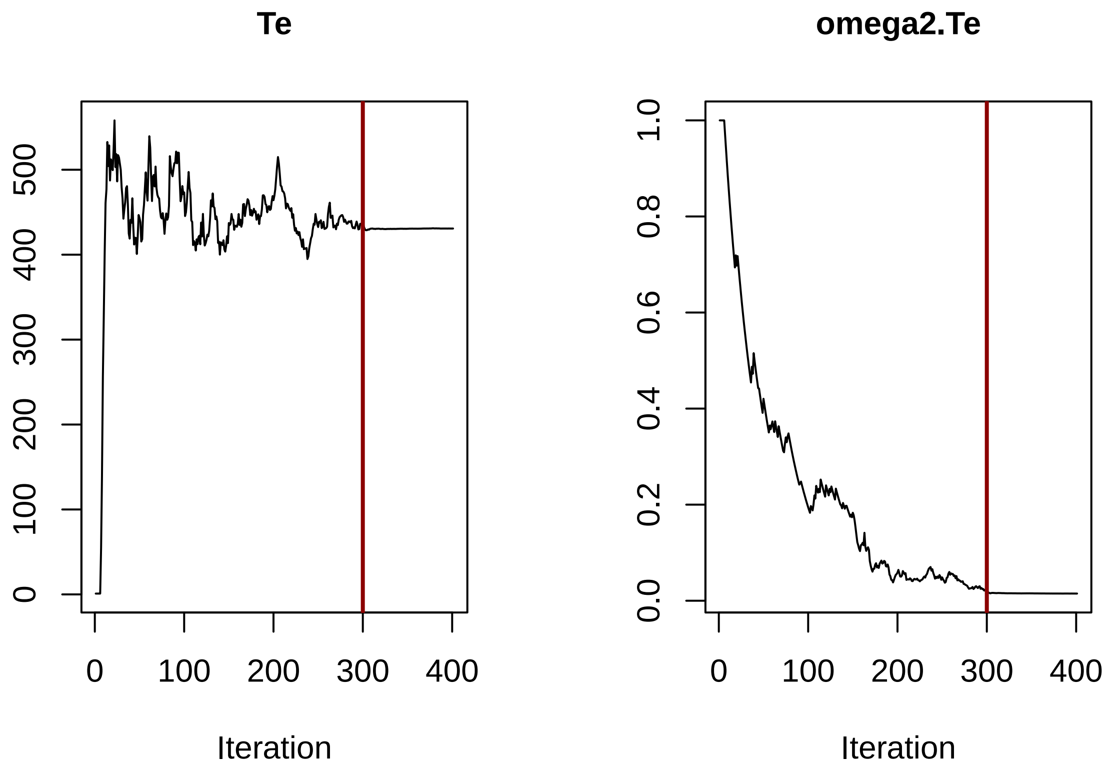
$$h(t) = \frac{1}{T_e \Gamma(k)} \left(\frac{t}{T_e}\right)^{(k-1)} \, e^{-\frac{t}{T_e}}$$

- log-logistic model:

$$h(t) = TODO$$

The code below fits all these models to the lung cancer data.

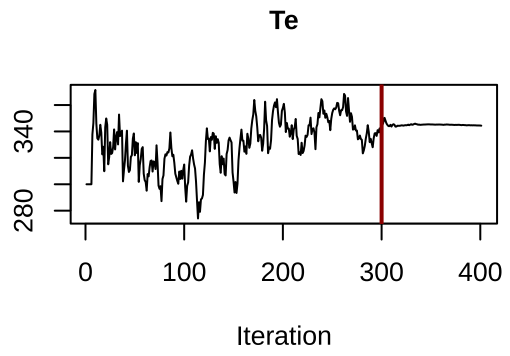
```r
# Exponential
exptte.model<-function(psi,id,xidep) {
  T<-xidep[,1]
  y<-xidep[,2] # events (1=event, 0=no event)
  cens<-which(xidep[,3]==1) # censoring times (subject specific)
  init <- which(T==0)
  Te <- psi[id,1] # Parameters of the Weibull model
  Nj <- length(T)

  ind <- setdiff(1:Nj, append(init,cens)) # indices of events
  hazard <- (1/Te) # H'
  H <- (T/Te) #  H= -ln(S)
  logpdf <- rep(0,Nj) # ln(l(T=0))=0
  logpdf[cens] <- -H[cens] + H[cens-1] # ln(l(T=censoring time))=ln(S)=-H
  logpdf[ind] <- -H[ind] + H[ind-1] + log(hazard[ind]) # ln(l(T=event time))=ln(S)+ln(h)

  return(logpdf)
}

saemix.model.exp<-saemixModel(model=exptte.model,description="Exponential TTE model",modeltype="likelih
  psi0=matrix(c(1),ncol=1,byrow=TRUE,dimnames=list(NULL,  c("Te"))),
  transform.par=c(1),covariance.model=matrix(c(1),ncol=1, byrow=TRUE), verbose=FALSE)
saemix.options<-list(seed=632545,save=FALSE,save.graphs=FALSE, displayProgress=FALSE, print=FALSE)
exptte.fit<-saemix(saemix.model.exp,saemix.data.contPH,saemix.options)
plot(exptte.fit, plot.type="convergence")
```

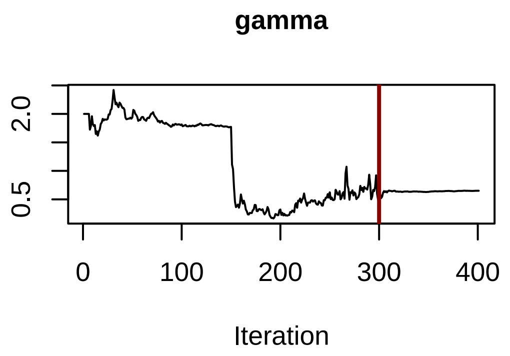


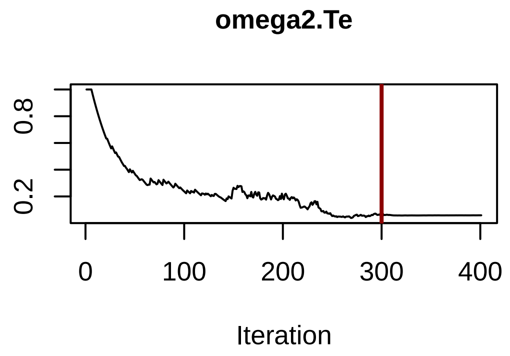
```r
print(exptte.fit)
```

```
## Nonlinear mixed-effects model fit by the SAEM algorithm
## -----------------------------------
## ----          Data             ----
## -----------------------------------
## Object of class SaemixData
##     longitudinal data for use with the SAEM algorithm
## Dataset lung1
##     Structured data: status ~ time + status + cens | id
##     X variable for graphs: time (days)
##     covariates: sex (), ph.ecog (-), ph.karno (%), pat.karno (%), age (yr)
##       reference class for covariate sex :  0
## Dataset characteristics:
##     number of subjects:     225
##     number of observations: 450
##     average/min/max nb obs: 2.00  /  2  /  2
## First 10 lines of data:
##    id time status cens status.1 sex ph.ecog ph.karno pat.karno age mdv cens.1
## 1   1    0      0    0        0   0       1       90       100  74   0      0
## 2   1  306      1    0        1   0       1       90       100  74   0      0
## 3   2    0      0    0        0   0       0       90        90  68   0      0
## 4   2  455      1    0        1   0       0       90        90  68   0      0
## 5   3    0      0    0        0   0       0       90        90  56   0      0
## 6   3 1010      0    1        0   0       0       90        90  56   0      0
## 7   4    0      0    0        0   0       1       90        60  57   0      0
## 8   4  210      1    0        1   0       1       90        60  57   0      0
## 9   5    0      0    0        0   0       0      100        90  60   0      0
## 10  5  883      1    0        1   0       0      100        90  60   0      0
##    occ ytype
## 1    1     1
## 2    1     1
## 3    1     1
## 4    1     1
## 5    1     1
## 6    1     1
## 7    1     1
## 8    1     1
## 9    1     1
## 10   1     1
## -----------------------------------
## ----          Model            ----
## -----------------------------------
## Nonlinear mixed-effects model
##   Model function:  Exponential TTE model
##   Model type:  likelihood
## function(psi,id,xidep) {
##   T<-xidep[,1]
##   y<-xidep[,2] # events (1=event, 0=no event)
##   cens<-which(xidep[,3]==1) # censoring times (subject specific)
##   init <- which(T==0)
##   Te <- psi[id,1] # Parameters of the Weibull model
##   Nj <- length(T)
##
##   ind <- setdiff(1:Nj, append(init,cens)) # indices of events
##   hazard <- (1/Te) # H'
##   H <- (T/Te) #  H= -ln(S)
##   logpdf <- rep(0,Nj) # ln(l(T=0))=0
##   logpdf[cens] <- -H[cens] + H[cens-1] # ln(l(T=censoring time))=ln(S)=-H
##   logpdf[ind] <- -H[ind] + H[ind-1] + log(hazard[ind]) # ln(l(T=event time))=ln(S)+ln(h)
##
##   return(logpdf)
## }
## <bytecode: 0x555e0f0c0178>
##   Nb of parameters: 1
##       parameter names:  Te
##       distribution:
##      Parameter Distribution Estimated
## [1,] Te        log-normal   Estimated
##   Variance-covariance matrix:
##    Te
## Te  1
##     No covariate in the model.
##     Initial values
##              Te
## Pop.CondInit  1
## ----------------------------------
## ----    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:  632545
##     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:  FALSE
##         save the graphs to files:  FALSE
## ----------------------------------------------------
## ----                  Results                   ----
## ----------------------------------------------------
## -----------------  Fixed effects  ------------------
## ----------------------------------------------------
##      Parameter Estimate SE CV(%)
## [1,] Te        431      57 13
## ----------------------------------------------------
## -----------  Variance of random effects  -----------
## ----------------------------------------------------
##    Parameter Estimate SE  CV(%)
## Te omega2.Te 0.015    0.3 2002
## ----------------------------------------------------
## ------  Correlation matrix of random effects  ------
## ----------------------------------------------------
##           omega2.Te
## omega2.Te 1
## ----------------------------------------------------
## ---------------  Statistical criteria  -------------
## ----------------------------------------------------
## Likelihood computed by linearisation
##       -2LL= 8604.435
##       AIC = 8610.435
##       BIC = 8620.683
##
## Likelihood computed by importance sampling
##       -2LL= 2286.805
##       AIC = 2292.805
##       BIC = 2303.053
## ----------------------------------------------------
```

```
# Gompertz

gomptte.model<-function(psi,id,xidep) {
  T<-xidep[,1]
  y<-xidep[,2] # events (1=event, 0=no event)
  cens<-which(xidep[,3]==1) # censoring times (subject specific)
  init <- which(T==0)
  Te <- psi[id,1] # Parameters of the Weibull model
  gamma <- psi[id,2]
  teprim <- Te/log(1+log(2)/gamma)
  Nj <- length(T)

  ind <- setdiff(1:Nj, append(init,cens)) # indices of events
  hazard <- (gamma/teprim)*exp(T/teprim) # h
  H <- gamma*(exp(T/teprim)-1) # H
  logpdf <- rep(0,Nj) # ln(l(T=0))=0
  logpdf[cens] <- -H[cens] + H[cens-1] # ln(l(T=censoring time))=ln(S)=-H
  logpdf[ind] <- -H[ind] + H[ind-1] + log(hazard[ind]) # ln(l(T=event time))=ln(S)+ln(h)
  return(logpdf)
}
saemix.model.gomp<-saemixModel(model=gomptte.model,description="Gompertz TTE model",modeltype="likeliho
  psi0=matrix(c(300,2),ncol=2,byrow=TRUE,dimnames=list(NULL,  c("Te","gamma"))),
  transform.par=c(1,1),covariance.model=matrix(c(1,0,0,0),ncol=2, byrow=TRUE), verbose=FALSE)
saemix.options<-list(seed=632545,save=FALSE,save.graphs=FALSE, displayProgress=FALSE, print=FALSE)
gomptte.fit<-saemix(saemix.model.gomp,saemix.data.contPH,saemix.options)
plot(gomptte.fit, plot.type="convergence")
```



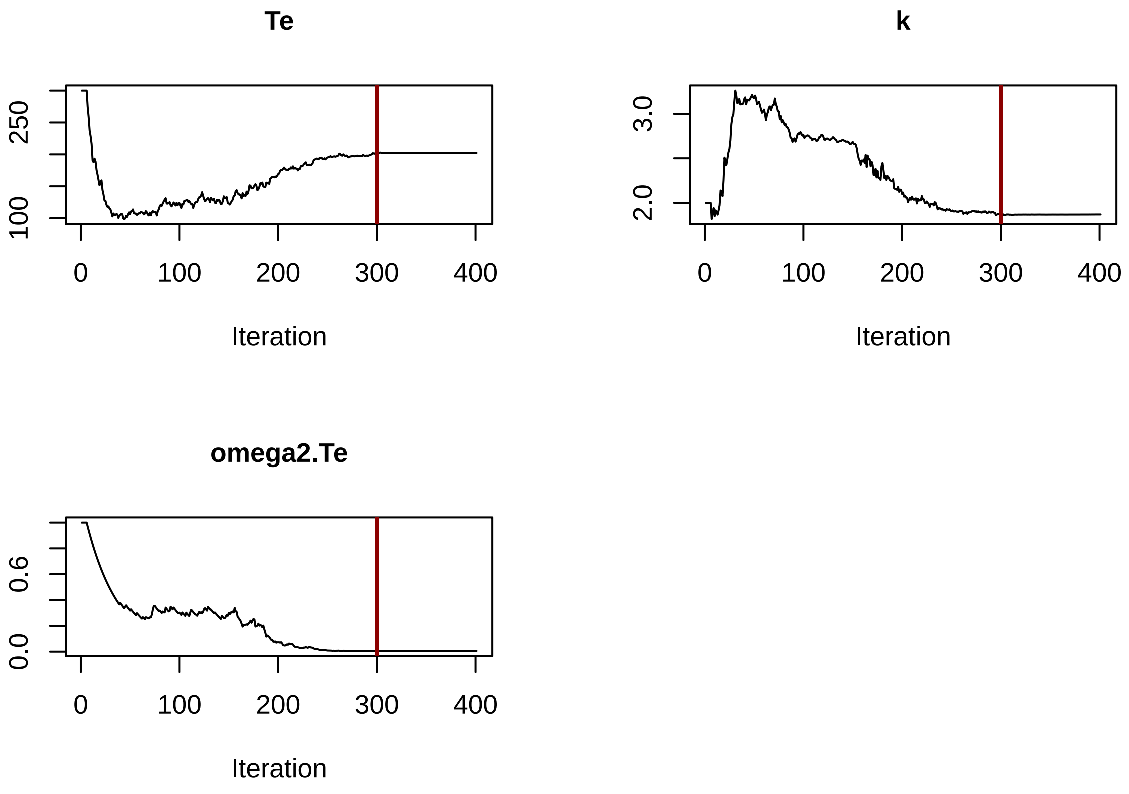




```r
print(gomptte.fit)
```

```
## Nonlinear mixed-effects model fit by the SAEM algorithm
## -----------------------------------
## ----          Data              ----
## -----------------------------------
## Object of class SaemixData
##     longitudinal data for use with the SAEM algorithm
## Dataset lung1
##     Structured data: status ~ time + status + cens | id
##     X variable for graphs: time (days)
##     covariates: sex (), ph.ecog (-), ph.karno (%), pat.karno (%), age (yr)
##       reference class for covariate sex :  0
## Dataset characteristics:
##     number of subjects:     225
##     number of observations: 450
##     average/min/max nb obs: 2.00  /  2  /  2
## First 10 lines of data:
##    id time status cens status.1 sex ph.ecog ph.karno pat.karno age mdv cens.1
## 1   1    0      0    0        0   0       1       90       100  74   0      0
## 2   1  306      1    0        1   0       1       90       100  74   0      0
## 3   2    0      0    0        0   0       0       90        90  68   0      0
## 4   2  455      1    0        1   0       0       90        90  68   0      0
## 5   3    0      0    0        0   0       0       90        90  56   0      0
## 6   3 1010      0    1        0   0       0       90        90  56   0      0
## 7   4    0      0    0        0   0       1       90        60  57   0      0
## 8   4  210      1    0        1   0       1       90        60  57   0      0
## 9   5    0      0    0        0   0       0      100        90  60   0      0
## 10  5  883      1    0        1   0       0      100        90  60   0      0
##    occ ytype
## 1    1     1
## 2    1     1
## 3    1     1
## 4    1     1
## 5    1     1
## 6    1     1
## 7    1     1
## 8    1     1
## 9    1     1
## 10   1     1
## ----------------------------------
## ----          Model            ----
## ----------------------------------
## Nonlinear mixed-effects model
##   Model function:  Gompertz TTE model
##   Model type:  likelihood
## function(psi,id,xidep) {
##   T<-xidep[,1]
##   y<-xidep[,2] # events (1=event, 0=no event)
##   cens<-which(xidep[,3]==1) # censoring times (subject specific)
##   init <- which(T==0)
##   Te <- psi[id,1] # Parameters of the Weibull model
##   gamma <- psi[id,2]
##   teprim <- Te/log(1+log(2)/gamma)
##   Nj <- length(T)
##
##   ind <- setdiff(1:Nj, append(init,cens)) # indices of events
##   hazard <- (gamma/teprim)*exp(T/teprim) # h
##   H <- gamma*(exp(T/teprim)-1) # H
##   logpdf <- rep(0,Nj) # ln(l(T=0))=0
##   logpdf[cens] <- -H[cens] + H[cens-1] # ln(l(T=censoring time))=ln(S)=-H
##   logpdf[ind] <- -H[ind] + H[ind-1] + log(hazard[ind]) # ln(l(T=event time))=ln(S)+ln(h)
##   return(logpdf)
## }
## <bytecode: 0x555e0f374528>
##   Nb of parameters: 2
##       parameter names:  Te gamma
##       distribution:
##      Parameter Distribution Estimated
## [1,] Te        log-normal   Estimated
## [2,] gamma     log-normal   Estimated
##   Variance-covariance matrix:
##       Te gamma
## Te     1     0
## gamma  0     0
##     No covariate in the model.
##     Initial values
##               Te gamma
## Pop.CondInit 300     2
## ----------------------------------
## ----    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:  632545
##     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:  FALSE
##         save the graphs to files:  FALSE
## ----------------------------------------------------
## ----                  Results                   ----
## ----------------------------------------------------
## -----------------  Fixed effects  ------------------
## ----------------------------------------------------
##      Parameter Estimate SE    CV(%)
## [1,] Te        344.36   36.46 11
## [2,] gamma       0.65    0.34 53
## ----------------------------------------------------
## -----------  Variance of random effects  -----------
## ----------------------------------------------------
##    Parameter Estimate SE  CV(%)
## Te omega2.Te 0.059    0.2 344
## ----------------------------------------------------
## ------  Correlation matrix of random effects  ------
## ----------------------------------------------------
##           omega2.Te
## omega2.Te 1
## ----------------------------------------------------
## ---------------  Statistical criteria  -------------
## ----------------------------------------------------
## Likelihood computed by linearisation
##       -2LL= 6374.505
##       AIC = 6382.505
##       BIC = 6396.169
##
## Likelihood computed by importance sampling
##       -2LL= 2270.115
##       AIC = 2278.115
##       BIC = 2291.779
## ----------------------------------------------------
```

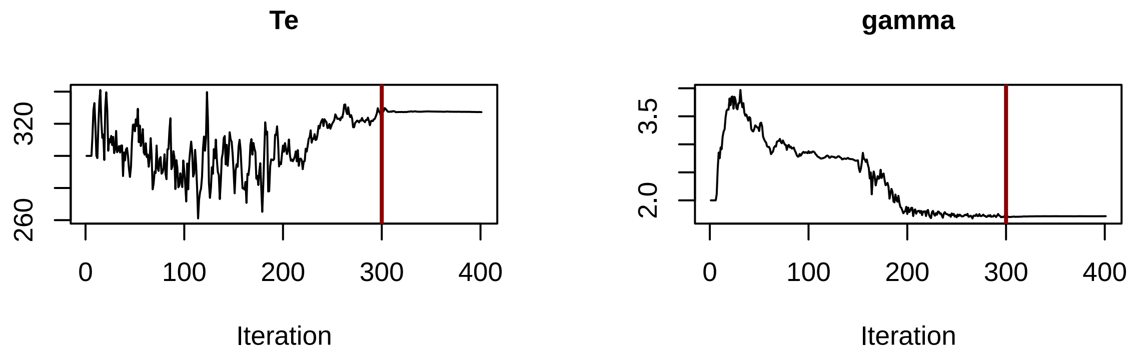
```
# Gamma
# incomplete gamma function for (x,a) : gamma(a) * pgamma(x, a, 1, lower = FALSE)

gammatte.model<-function(psi,id,xidep) {
  T<-xidep[,1]
  y<-xidep[,2] # events (1=event, 0=no event)
  cens<-which(xidep[,3]==1) # censoring times (subject specific)
  init <- which(T==0)
  Te <- psi[id,1] # Parameters of the Weibull model
  lambda <- psi[id,2]
  Nj <- length(T)

  ind <- setdiff(1:Nj, append(init,cens)) # indices of events
#  hazard <- (lambda/Te) * (lambda*T/Te)^(lambda-1) * exp(lambda*T/Te) / (gamma(lambda) - pgamma(lambda
  hazard <- (T/Te)^(lambda-1) * exp(-T/Te) /gamma(lambda) / Te
#  H <- pgamma(T/Te, lambda, 1, lower=FALSE) / gamma(lambda)
  H <- pgamma(T/Te, lambda) # incomplete gamma gammainc(x,a)=pgamma(x,a)*gamma(a) and H=gammainc(T/Te,
#  H <- (1-pgamma(T/Te, lambda,1, lower=FALSE))
  logpdf <- rep(0,Nj) # ln(l(T=0))=0
  logpdf[cens] <- -H[cens] + H[cens-1] # ln(l(T=censoring time))
  logpdf[ind] <- -H[ind] + H[ind-1] + log(hazard[ind]) # ln(l(T=event time))
  return(logpdf)
}
saemix.model.gamma<-saemixModel(model=gammatte.model,description="Gamma TTE model",modeltype="likelihoo
  psi0=matrix(c(300,2),ncol=2,byrow=TRUE,dimnames=list(NULL,  c("Te","k"))),
  transform.par=c(1,1),covariance.model=matrix(c(1,0,0,0),ncol=2, byrow=TRUE), verbose=FALSE)
saemix.options<-list(seed=632545,save=FALSE,save.graphs=FALSE, displayProgress=FALSE, print=FALSE)
gammatte.fit<-try(saemix(saemix.model.gamma,saemix.data.contPH,saemix.options))
plot(gammatte.fit, plot.type="convergence")
```

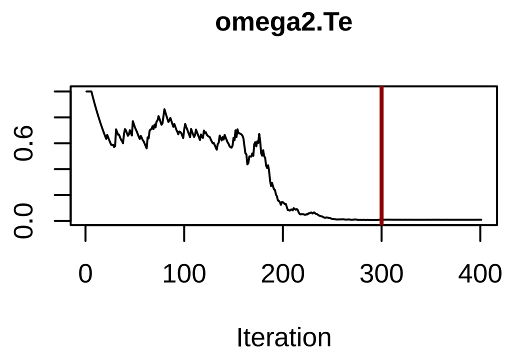


```
print(gammatte.fit)
```

```
## Nonlinear mixed-effects model fit by the SAEM algorithm
## -----------------------------------
## ----          Data              ----
## -----------------------------------
## Object of class SaemixData
##     longitudinal data for use with the SAEM algorithm
## Dataset lung1
##     Structured data: status ~ time + status + cens | id
##     X variable for graphs: time (days)
##     covariates: sex (), ph.ecog (-), ph.karno (%), pat.karno (%), age (yr)
##       reference class for covariate sex :  0
## Dataset characteristics:
##     number of subjects:     225
##     number of observations: 450
##     average/min/max nb obs: 2.00  /  2  /  2
## First 10 lines of data:
##    id time status cens status.1 sex ph.ecog ph.karno pat.karno age mdv cens.1
## 1   1    0      0    0        0   0       1       90       100  74   0      0
## 2   1  306      1    0        1   0       1       90       100  74   0      0
## 3   2    0      0    0        0   0       0       90        90  68   0      0
## 4   2  455      1    0        1   0       0       90        90  68   0      0
## 5   3    0      0    0        0   0       0       90        90  56   0      0
## 6   3 1010      0    1        0   0       0       90        90  56   0      0
## 7   4    0      0    0        0   0       1       90        60  57   0      0
## 8   4  210      1    0        1   0       1       90        60  57   0      0
## 9   5    0      0    0        0   0       0      100        90  60   0      0
## 10  5  883      1    0        1   0       0      100        90  60   0      0
##    occ ytype
## 1    1     1
## 2    1     1
## 3    1     1
## 4    1     1
## 5    1     1
## 6    1     1
## 7    1     1
## 8    1     1
## 9    1     1
## 10   1     1
## -----------------------------------
## ----          Model            ----
## -----------------------------------
## Nonlinear mixed-effects model
##   Model function:  Gamma TTE model
##   Model type:  likelihood
## function(psi,id,xidep) {
##   T<-xidep[,1]
##   y<-xidep[,2] # events (1=event, 0=no event)
##   cens<-which(xidep[,3]==1) # censoring times (subject specific)
##   init <- which(T==0)
##   Te <- psi[id,1] # Parameters of the Weibull model
##   lambda <- psi[id,2]
##   Nj <- length(T)
##
##   ind <- setdiff(1:Nj, append(init,cens)) # indices of events
## #  hazard <- (lambda/Te) * (lambda*T/Te)^(lambda-1) * exp(lambda*T/Te) / (gamma(lambda) - pgamma(lam
##   hazard <- (T/Te)^(lambda-1) * exp(-T/Te) /gamma(lambda) / Te
## #  H <- pgamma(T/Te, lambda, 1, lower=FALSE) / gamma(lambda)
##   H <- pgamma(T/Te, lambda) # incomplete gamma gammainc(x,a)=pgamma(x,a)*gamma(a) and H=gammainc(T/T
## #  H <- (1-pgamma(T/Te, lambda,1, lower=FALSE))
##   logpdf <- rep(0,Nj) # ln(l(T=0))=0
##   logpdf[cens] <- -H[cens] + H[cens-1] # ln(l(T=censoring time))
##   logpdf[ind] <- -H[ind] + H[ind-1] + log(hazard[ind]) # ln(l(T=event time))
##   return(logpdf)
## }
## <bytecode: 0x555e02325e80>
##   Nb of parameters: 2
##       parameter names:  Te k
##       distribution:
##      Parameter Distribution Estimated
## [1,] Te        log-normal   Estimated
## [2,] k         log-normal   Estimated
##   Variance-covariance matrix:
##    Te k
## Te  1 0
## k   0 0
##     No covariate in the model.
##     Initial values
##               Te k
## Pop.CondInit 300 2
## ----------------------------------
## ----    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:  632545
##     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:  FALSE
##         save the graphs to files:  FALSE
## ----------------------------------------------------
## ----                  Results                   ----
## ----------------------------------------------------
## -----------------  Fixed effects  ------------------
## ----------------------------------------------------
##      Parameter Estimate SE    CV(%)
## [1,] Te        202.2    60.67 30
## [2,] k           1.9     0.42 22
## ----------------------------------------------------
## -----------  Variance of random effects  -----------
## ----------------------------------------------------
##    Parameter Estimate SE   CV(%)
## Te omega2.Te 0.0052   0.16 3118
## ----------------------------------------------------
## ------  Correlation matrix of random effects  ------
## ----------------------------------------------------
##           omega2.Te
## omega2.Te 1
## ----------------------------------------------------
## ---------------  Statistical criteria  -------------
## ----------------------------------------------------
## Likelihood computed by linearisation
##       -2LL= 5132.122
##       AIC = 5140.122
##       BIC = 5153.786
##
## Likelihood computed by importance sampling
##       -2LL= 2356.678
##       AIC = 2364.678
##       BIC = 2378.342
## ----------------------------------------------------
```

```
# Log-logistic
logis.model<-function(psi,id,xidep) {
  T<-xidep[,1]
  y<-xidep[,2] # events (1=event, 0=no event)
  cens<-which(xidep[,3]==1) # censoring times (subject specific)
  init <- which(T==0)
  Te <- psi[id,1] # Parameters of the Weibull model
  gamma <- psi[id,2]
  Nj <- length(T)

  ind <- setdiff(1:Nj, append(init,cens)) # indices of events
  hazard <- (gamma/Te)*(T/Te)^(gamma-1) /(1+(T/Te)^gamma) # H'
  H <- log(1+(T/Te)^gamma) # H= -ln(S)
  logpdf <- rep(0,Nj) # ln(l(T=0))=0
  logpdf[cens] <- -H[cens] + H[cens-1] # ln(l(T=censoring time))=ln(S)=-H
  logpdf[ind] <- -H[ind] + H[ind-1] + log(hazard[ind]) # ln(l(T=event time))=ln(S)+ln(h)
  return(logpdf)
}

saemix.model.logis<-saemixModel(model=logis.model,description="Log-logistic TTE model",modeltype="likeli
  psi0=matrix(c(300,2),ncol=2,byrow=TRUE,dimnames=list(NULL,  c("Te","gamma"))),
  transform.par=c(1,1),covariance.model=matrix(c(1,0,0,0),ncol=2, byrow=TRUE), verbose=FALSE)
saemix.options<-list(seed=632545,save=FALSE,save.graphs=FALSE, displayProgress=FALSE, print=FALSE)
logistte.fit<-saemix(saemix.model.logis,saemix.data.contPH,saemix.options)
plot(logistte.fit, plot.type="convergence")
```





```r
print(logistte.fit)
```

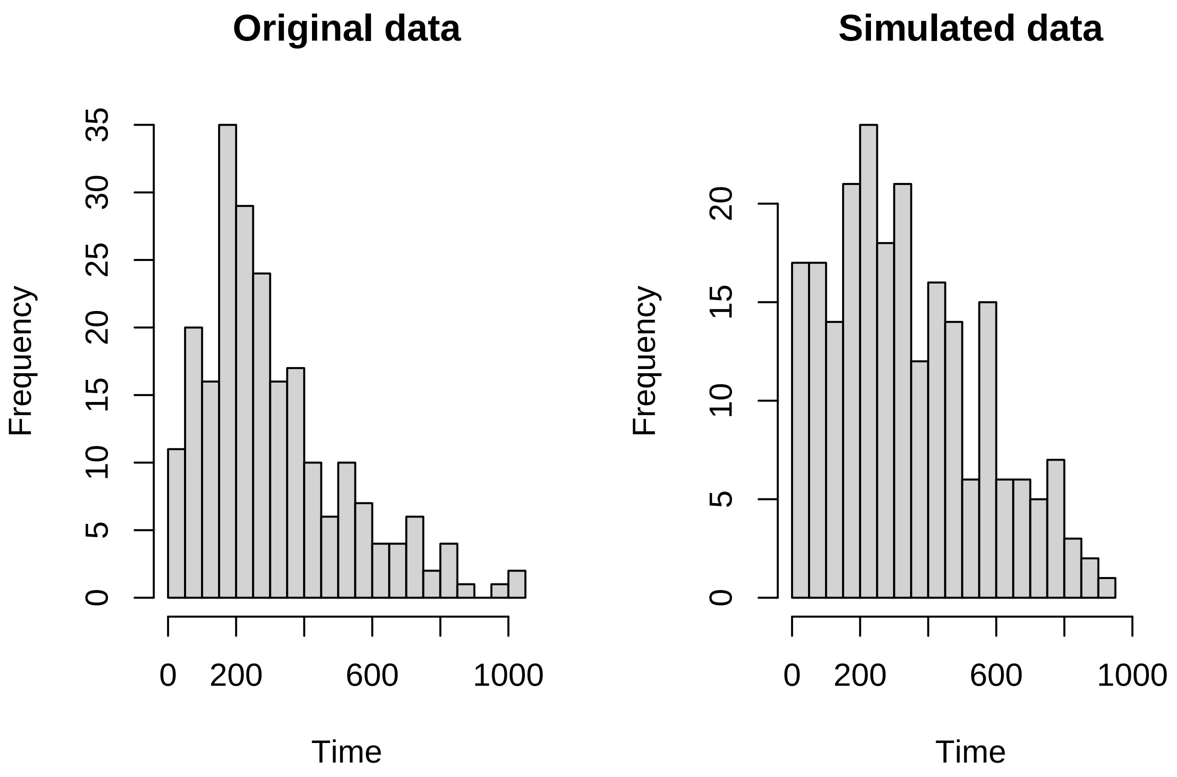
```
## Nonlinear mixed-effects model fit by the SAEM algorithm
## -----------------------------------
## ----          Data              ----
## -----------------------------------
## Object of class SaemixData
##     longitudinal data for use with the SAEM algorithm
## Dataset lung1
##     Structured data: status ~ time + status + cens | id
##     X variable for graphs: time (days)
##     covariates: sex (), ph.ecog (-), ph.karno (%), pat.karno (%), age (yr)
##       reference class for covariate sex :  0
## Dataset characteristics:
##     number of subjects:     225
##     number of observations: 450
##     average/min/max nb obs: 2.00  /  2  /  2
## First 10 lines of data:
##    id time status cens status.1 sex ph.ecog ph.karno pat.karno age mdv cens.1
## 1   1    0      0    0        0   0       1       90       100  74   0      0
## 2   1  306      1    0        1   0       1       90       100  74   0      0
## 3   2    0      0    0        0   0       0       90        90  68   0      0
## 4   2  455      1    0        1   0       0       90        90  68   0      0
## 5   3    0      0    0        0   0       0       90        90  56   0      0
## 6   3 1010      0    1        0   0       0       90        90  56   0      0
## 7   4    0      0    0        0   0       1       90        60  57   0      0
## 8   4  210      1    0        1   0       1       90        60  57   0      0
## 9   5    0      0    0        0   0       0      100        90  60   0      0
## 10  5  883      1    0        1   0       0      100        90  60   0      0
##    occ ytype
## 1    1     1
## 2    1     1
## 3    1     1
## 4    1     1
## 5    1     1
## 6    1     1
## 7    1     1
## 8    1     1
## 9    1     1
## 10   1     1
## ----------------------------------
## ----          Model           ----
## ----------------------------------
## Nonlinear mixed-effects model
##   Model function:  Log-logistic TTE model
##   Model type:  likelihood
## function(psi,id,xidep) {
##   T<-xidep[,1]
##   y<-xidep[,2] # events (1=event, 0=no event)
##   cens<-which(xidep[,3]==1) # censoring times (subject specific)
##   init <- which(T==0)
##   Te <- psi[id,1] # Parameters of the Weibull model
##   gamma <- psi[id,2]
##   Nj <- length(T)
##
##   ind <- setdiff(1:Nj, append(init,cens)) # indices of events
##   hazard <- (gamma/Te)*(T/Te)^(gamma-1) /(1+(T/Te)^gamma) # H'
##   H <- log(1+(T/Te)^gamma) # H= -ln(S)
##   logpdf <- rep(0,Nj) # ln(l(T=0))=0
##   logpdf[cens] <- -H[cens] + H[cens-1] # ln(l(T=censoring time))=ln(S)=-H
##   logpdf[ind] <- -H[ind] + H[ind-1] + log(hazard[ind]) # ln(l(T=event time))=ln(S)+ln(h)
##   return(logpdf)
## }
## <bytecode: 0x555e021dc490>
##   Nb of parameters: 2
##       parameter names:  Te gamma
##       distribution:
##      Parameter Distribution Estimated
## [1,] Te        log-normal   Estimated
## [2,] gamma     log-normal   Estimated
##   Variance-covariance matrix:
##       Te gamma
## Te     1     0
## gamma  0     0
##     No covariate in the model.
##     Initial values
##              Te gamma
## Pop.CondInit 300     2
## ----------------------------------
## ----    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:  632545
##     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:  FALSE
##         save the graphs to files:  FALSE
## ----------------------------------------------------
## ----                  Results                   ----
## ----------------------------------------------------
## -----------------  Fixed effects  ------------------
## ----------------------------------------------------
##      Parameter Estimate SE    CV(%)
## [1,] Te        327.2    41.14 13
## [2,] gamma       1.7     0.24 14
## ----------------------------------------------------
## -----------  Variance of random effects  -----------
## ----------------------------------------------------
##    Parameter Estimate SE   CV(%)
## Te omega2.Te 0.0089   0.17 1918
## ----------------------------------------------------
## ------  Correlation matrix of random effects  ------
## ----------------------------------------------------
##           omega2.Te
## omega2.Te 1
## ----------------------------------------------------
## ---------------  Statistical criteria  -------------
## ----------------------------------------------------
## Likelihood computed by linearisation
##       -2LL= 5275.294
##       AIC = 5283.294
##       BIC = 5296.959
##
## Likelihood computed by importance sampling
##       -2LL= 2284.571
##       AIC = 2292.571
##       BIC = 2306.235
## ----------------------------------------------------
```

Comparing the models: Gompertz and Weibull have almost the same BIC. The diagnostic plots are nearly identical, with a slightly better fit towards the end with the Weibull model.

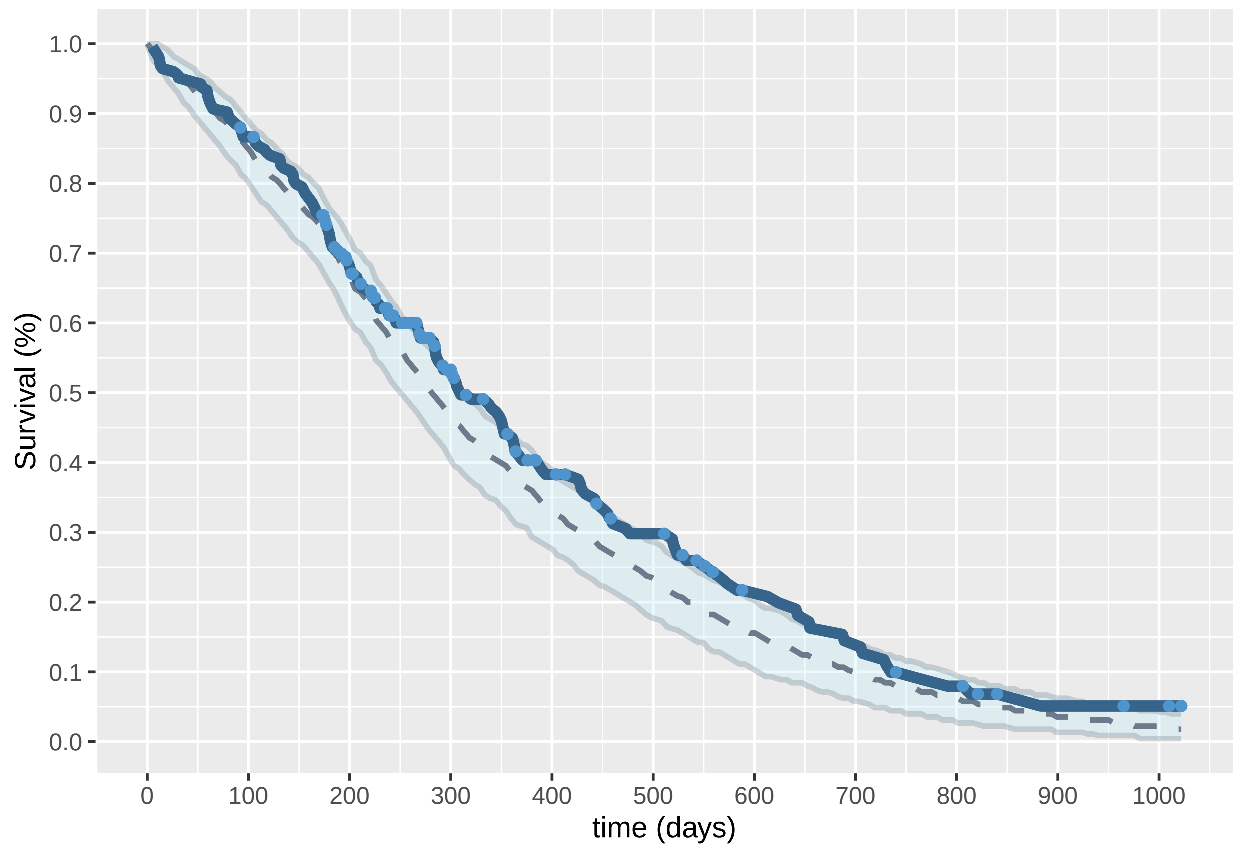
```r
# Table comparing the models
restte<-data.frame(Model=c("Exponential","Weibull","Gompertz","Gamma","Log-logistic"),
           BIC=c(BIC(exptte.fit),BIC(tte.fit), BIC(gomptte.fit), BIC(gammatte.fit), BIC(logistte.fit)))
print(restte)
```

```
##          Model      BIC
## 1  Exponential 2303.053
## 2      Weibull 2291.021
## 3     Gompertz 2291.779
## 4        Gamma 2378.342
## 5 Log-logistic 2306.235
```

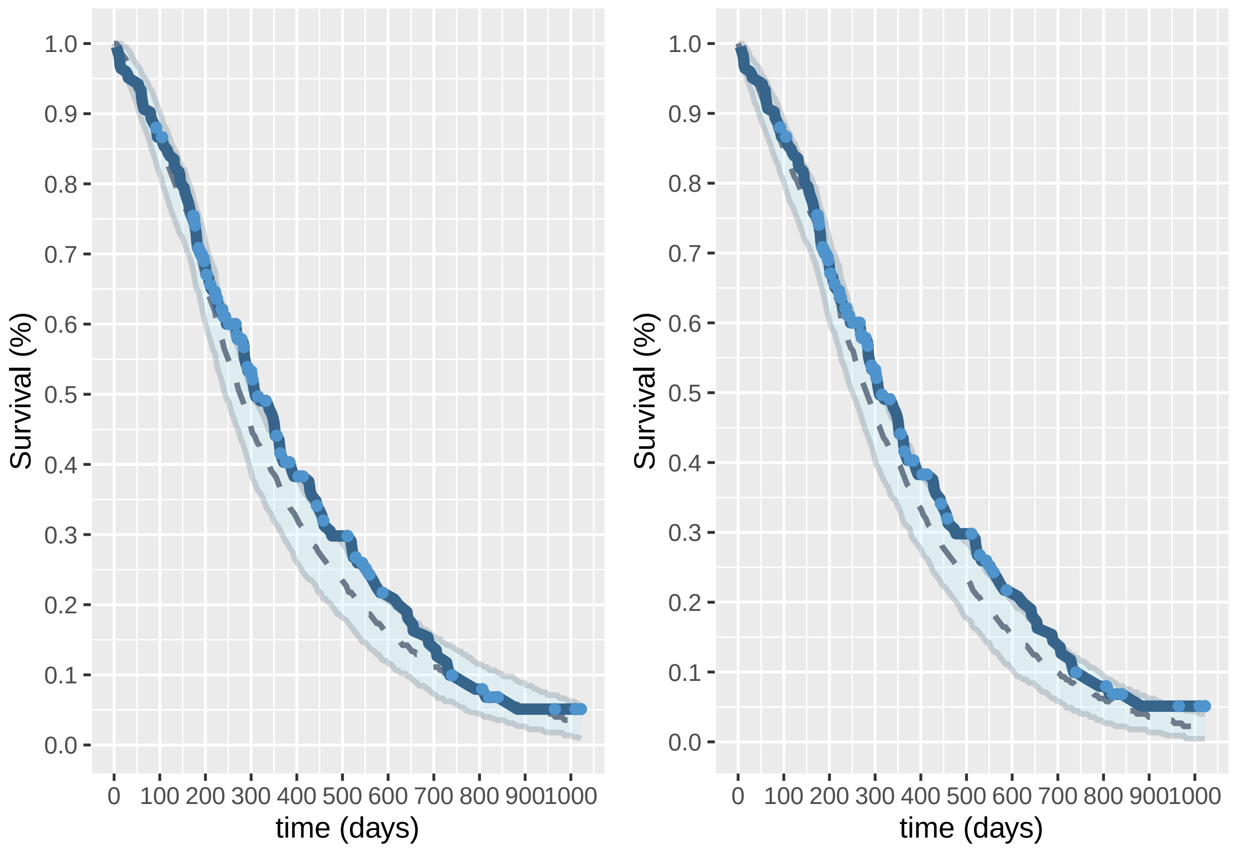
```
# Simulate events based on the observed individual censoring time
simulateGompertzTTE <- function(psi,id,xidep) {
  T<-xidep[,1]
  y<-xidep[,2] # events (1=event, 0=no event)
  delta <- xidep[,3] # censoring indicator
  cens<-which(delta==1) # censoring times (subject specific)
  tmax <- max(T[cens]) # maximum censoring time observed in dataset
  init <- which(T==0)
  Te <- psi[,1] # Parameters of the Weibull model
  gamma <- psi[,2]
  teprim <- Te/log(1+log(2)/gamma)

  Nj <- length(T)
  ind <- setdiff(1:Nj, append(init,cens)) # indices of events
  tevent<-T
  Vj<-runif(dim(psi)[1])
  tsim<-teprim*log(1-log(Vj)/gamma) #   events
  tevent[T>0]<-tsim
  tevent[delta==1 & tevent>T] <- T[delta==1 & tevent>T] # subject-specific censoring time
#  tevent[delta==0 & tevent>tmax] <- tmax # censoring to tmax (for subjects who experienced an event)
#  tevent[tevent[dead]>tmax] <- tmax # for subjects who initially experienced the event, use maximal ce
  return(tevent)
}

# Checking the simulation function
xidep1<-saemix.data.contPH@data[,saemix.data.contPH@name.predictors]
nsuj<-saemix.data.contPH@N
psiM<-data.frame(Te=rnorm(nsuj, mean=gomptte.fit@results@fixed.effects[1], sd=2), gamma=gomptte.fit@resu
id1<-rep(1:nsuj, each=2)
simtime<-simulateGompertzTTE(psiM, id1, xidep1)

par(mfrow=c(1,2))
hist(saemix.data.contPH@data$time[saemix.data.contPH@data$time>0], breaks=30, xlim=c(0,1050),xlab="Time
hist(simtime[simtime>0], breaks=30, xlim=c(0,1050), xlab="Time", main="Simulated data")
```



```r
gomptte.fit@model@simulate.function<-simulateGompertzTTE
simgomptte.fit <- simulateDiscreteSaemix(gomptte.fit, nsim=500)

gpl2 <- discreteVPC(simgomptte.fit, outcome="TTE")
plot(gpl2)
```



```
grid.arrange(gpl,gpl2, nrow=1)
```



#### Covariate model

The following section applies a stepwise procedure to test covariates with the selected model (here the Weibull model), making use of the BIC criterion developed by Delattre et al (2014). The final model includes an effect of sex, ECOG assessement and patient Karnofsky score.

The algorithm also tests different covariance structures for the model. Here, this may not be pertinent given the population can only experience a single event, therefore variability cannot really be identified.

```
# Toggle to TRUE to run (takes a while)
if(FALSE)
  covtte.fit <- step.saemix(tte.fit, direction="both")

# Covariate model

# Covariate model with only sex and ECOG score
```

SE via bootstrap

RTTE model

In this section we simulate repeated time-to-event data from a Weibull model and fit the dataset obtained. To simulate from a RTTE model, we simulate repeated events starting from the previous one using the inverse CDF technique. Because we don't know in advance the number of events in each subject, we lose the efficient vectorisation from **R** and this function can be considerably slower than the single event TTE.

The simulation function now becomes:

$$T = T_e \left(-\ln(V) + \left(\frac{T_{j-1}}{T_e} \right)^{\gamma} \right)^{1/\gamma}$$

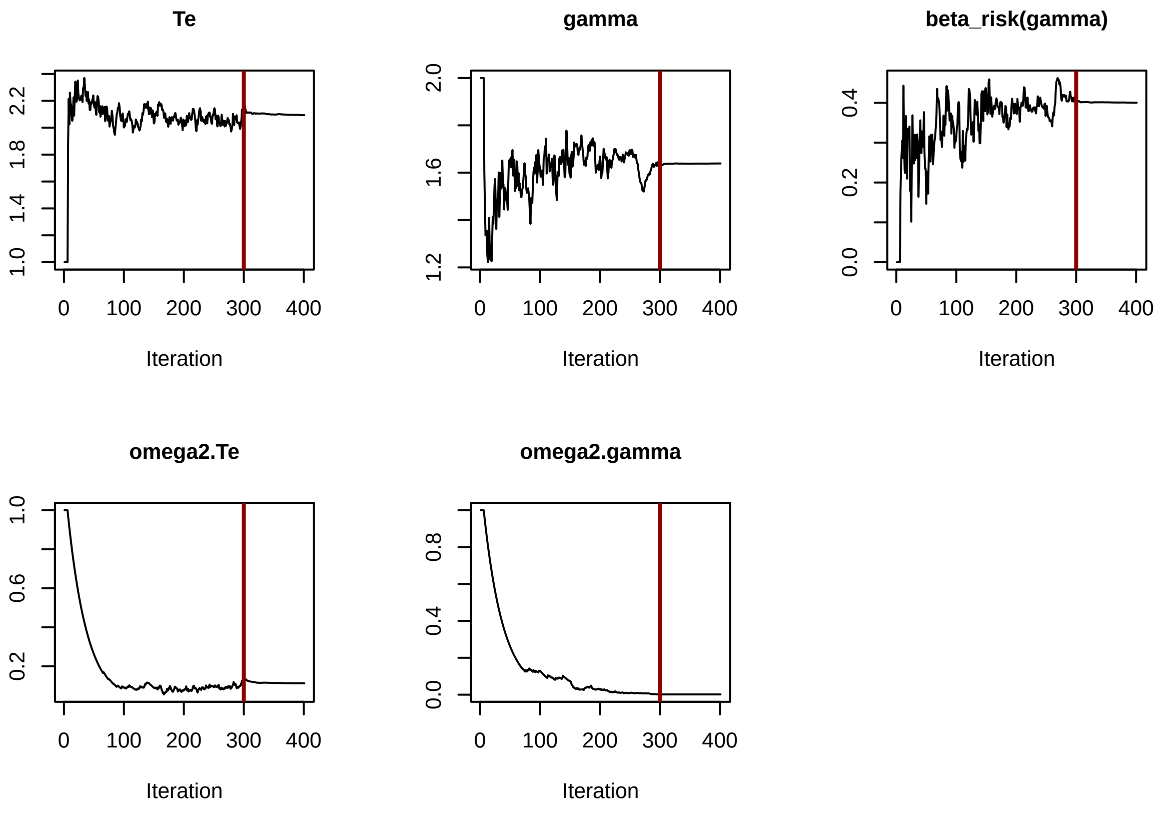
where T_{j-1} is the time of the last event and T the time of the next event (T_j).

```
# Simulating RTTE data by simulating from U(0,1) and inverting the cdf
simul.rtte.unif<-function(psi) { # xidep, id not important, we only use psi
  censoringtime <- 3
  maxevents <- 30
  Te <- psi[,1]
  gamma <- psi[,2]
  simdat<-NULL
  N<-nrow(psi)
  for(i in 1:N) {
    eventTimes<-c(0)
    T<-0
    Vj<-runif(1)
    #    T <- (-log(Vj)*Te[i])^(gamma[i])
    T<-Te[i]*(-log(Vj))^(1/gamma[i])
    nev<-0
    while (T < censoringtime & nev<maxevents){
      eventTimes <- c(eventTimes, T)
      nev<-nev+1
      Vj<-runif(1)
      #      T <- T+(-log(Vj)*Te[i])^(gamma[i])
      #      T<-(-log(Vj)*Te[i] + T^(1/gamma[i]))^(gamma[i])
      T<-Te[i]*(-log(Vj) + (T/Te[i])^(gamma[i]))^(1/gamma[i])
    }
    if(nev==maxevents) {
      message("Reached maximum number of events\n")
    }
    eventTimes<-c(eventTimes, censoringtime)
    cens<-rep(1,length(eventTimes))
    cens[1]<-cens[length(cens)]<-0
    simdat<-rbind(simdat,
                  data.frame(id=i, T=eventTimes, status=cens))
  }
  return(simdat)
}

# Subjects
set.seed(12345)
param<-c(2, 1.5, 0.5)
# param<-c(4, 1.2, 0.3)
omega<-c(0.25,0.25)
nsuj<-200
risk<-rep(0,nsuj)
risk[(nsuj/2+1):nsuj]<-1
psiM<-data.frame(Te=param[1]*exp(rnorm(nsuj,sd=omega[1])), gamma=param[2]*exp(param[3]*risk+rnorm(nsuj,s
simdat <- simul.rtte.unif(psiM)
```

```
## Reached maximum number of events
```

```
simdat$risk<-as.integer(simdat$id>(nsuj/2))
```

```
saemix.data<-saemixData(name.data=simdat, name.group=c("id"), name.predictors=c("T"), name.response="sta

rtte.model<-function(psi,id,xidep) {
  T<-xidep[,1]
  N <- nrow(psi) # nb of subjects
  Nj <- length(T) # nb of events (including 0 and censoring times)
  # censoringtime = 6
  censoringtime = max(T) # same censoring for everyone
  Te <- psi[id,1]
  gamma <- psi[id,2]
  tinit <- which(T==0) # indices of beginning of observation period
  tcens <- which(T==censoringtime) # indices of censored events
  tevent <- setdiff(1:Nj, append(tinit,tcens)) # indices of non-censored event times
  hazard <- (gamma/Te)*(T/Te)^(gamma-1)
  H <- (T/Te)^gamma
  logpdf <- rep(0,Nj)
  logpdf[tcens] <- -H[tcens] + H[tcens-1]
  logpdf[tevent] <- -H[tevent] + H[tevent-1] + log(hazard[tevent])
  return(logpdf)
}

saemix.model.base<-saemixModel(model=rtte.model,description="Repeated TTE model",modeltype="likelihood"
                               psi0=matrix(c(1,2),ncol=2,byrow=TRUE,dimnames=list(NULL,  c("Te","gamma"
                               transform.par=c(1,1),covariance.model=matrix(c(1,0,0,1),ncol=2, byrow=TRU
saemix.model<-saemixModel(model=rtte.model,description="Repeated TTE model",modeltype="likelihood",
                          psi0=matrix(c(1,2),ncol=2,byrow=TRUE,dimnames=list(NULL,  c("Te","gamma"))),
                          transform.par=c(1,1),covariate.model=matrix(c(0,1),ncol=2),
                          covariance.model=matrix(c(1,0,0,1),ncol=2, byrow=TRUE), verbose=FALSE)
saemix.options<-list(seed=632545,save=FALSE,save.graphs=FALSE, fim=FALSE, displayProgress=FALSE, print=
rtte.fit<-saemix(saemix.model,saemix.data,saemix.options)
plot(rtte.fit, plot.type="convergence")
```



```r
print(rtte.fit@results)
```

```
## ----------------------------------------------------
## -----------------  Fixed effects  ------------------
## ----------------------------------------------------
##      Parameter        Estimate
## [1,] Te               2.1
## [2,] gamma            1.6
## [3,] beta_risk(gamma) 0.4
## ----------------------------------------------------
## -----------  Variance of random effects  -----------
## ----------------------------------------------------
##       Parameter    Estimate
## Te    omega2.Te    0.1125
## gamma omega2.gamma 0.0015
## ----------------------------------------------------
## ------  Correlation matrix of random effects  ------
## ----------------------------------------------------
##              omega2.Te omega2.gamma
## omega2.Te    1         0
## omega2.gamma 0         1
## ----------------------------------------------------
## ---------------  Statistical criteria  -------------
## ----------------------------------------------------
##
## Likelihood computed by importance sampling
##       -2LL= 690.2485
##       AIC = 702.2485
##       BIC = 722.0384
## ----------------------------------------------------
```

Work in progress: currently, no diagnostic plots available for RTTE, stay tuned for progress.

Statistical model A nice review of the more frequent hazard functions used in parametric models of TTE data has recently been van Wijk and Simonsson (*CPT:PSP* 2022), including a Shiny app to explore their shape and how to set initial parameters. These models are very sensitive to the initial parameter estimates and their variance.

References

Comets E, Rodrigues C, Jullien V, Ursino M (2021). Conditional non-parametric bootstrap for non-linear mixed effect models. *Pharmaceutical Research*, 38: 1057-66.

Delattre M, Lavielle M, Poursat MA (2014) A note on BIC in mixed effects models. *Electronic Journal of Statistics* 8(1) p. 456-475

Delattre M, Poursat MA (2017) BIC strategies for model choice in a population approach. (arXiv:1612.02405)

Keizer R (2021). vpc: Create Visual Predictive Checks. *R package* version 1.2.2. https://CRAN.R-project.org/package=vpc

Loprinzi et al. (1994). Prospective evaluation of prognostic variables from patient-completed questionnaires. North Central Cancer Treatment Group. *Journal of Clinical Oncology : Official Journal of the American Society of Clinical Oncology*, 12(3), 601–607.

Morina D, Navarro A (2014). The R package survsim for the simulation of simple and complex survival Data. *Journal of Statistical Software*, 59(2), 1–20.

Ueckert S, Mentré F (2017). A new method for evaluation of the Fisher information matrix for discrete mixed effect models using Monte Carlo sampling and adaptive Gaussian quadrature. *Computational Statistics and Data Analysis*, 111: 203-19. 10.1016/j.csda.2016.10.011

van Wijk R, Simonsson U (2022). Finding the right hazard function for time-to-event modeling: A tutorial and Shiny application. *Clinical Pharmacokinetics and Therapeutics: Pharmacometrics and Systems Pharmacology*