

Testing examples in saemix 3.2 - discrete models

Emmanuelle

10/11/2022

Objective

Check saemix for discrete data models

Setup

- set up work directories
- two versions toggled by testMode
 - if testMode is FALSE, load the functions in R
 - if testMode is TRUE, load the library in a dev_mode environment
- aim: check the examples used in the online documentation
 - all examples must run without error

```
if(testMode) cat("Testing package\n") else cat("Loading functions\n")
```

```
## Testing package
```

Testing library

Binary response model

- Toenail data
 - using the full model with 2 random effects (better than with only random effect on intercept according to AIC/BIC)
 - quick diagnostics using a simulation function
- **TODO**
 - add diagnostics (npd-categorical ?)
 - maybe check SE's with package by S. Ueckert

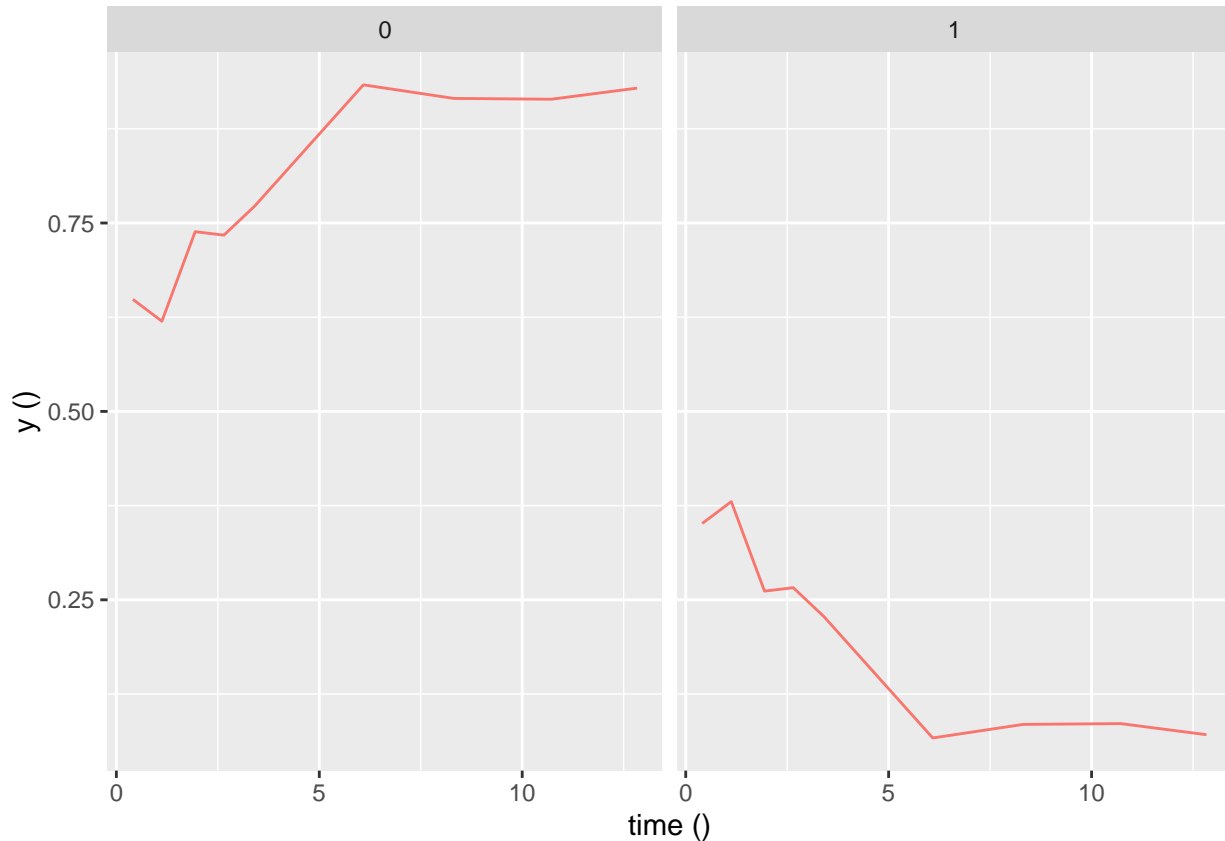
```
if(testMode)
  data(toenail.saemix) else
  toenail.saemix<-read.table(file.path(datDir, "toenail.saemix.tab"), header=TRUE)

saemix.data<-saemixData(name.data=toenail.saemix,name.group=c("id"),name.predictors=c("time","y"), name
                        name.covariates=c("treatment"),name.X=c("time"))
```

```
##
##
## The following SaemixData object was successfully created:
##
## Object of class SaemixData
##   longitudinal data for use with the SAEM algorithm
## Dataset toenail.saemix
```

```
##      Structured data: y ~ time + y | id
##      X variable for graphs: time ()
##      covariates: treatment (-)
##      reference class for covariate treatment : 0
```

```
plotDiscreteData(saemix.data, outcome='binary')
```



```
# Explore data
toe1 <- toenail.saemix %>%
  group_by(visit, treatment) %>%
  summarise(nev = sum(y), n=n()) %>%
  mutate(freq = nev/n, sd=sqrt((1-nev/n)/nev)) %>%
  mutate(lower=freq-1.96*sd, upper=freq+1.96*sd)
```

```
## `summarise()` has grouped output by 'visit'. You can override using the
## `.groups` argument.
```

```
toe1$lower[toe1$lower<0] <-0 # we should use a better approximation for CI
toe1$treatment <- factor(toe1$treatment, labels=c("A","B"))
```

```
plot1<-ggplot(toe1, aes(x=visit, y=freq, group=treatment)) + geom_line(aes(colour=treatment)) +
  geom_point(aes(colour=treatment)) +
  geom_ribbon(aes(ymin=lower, ymax=upper, fill=treatment), alpha=0.2) +
  ylim(c(0,1)) + theme_bw() + theme(legend.position = "top") +
  xlab("Visit number") + ylab("Observed frequency of infection")
```

```
# saemix model
binary.model<-function(psi,id,xidep) {
```

```

tim<-xidep[,1]
y<-xidep[,2]
inter<-psi[id,1]
slope<-psi[id,2]
logit<-inter+slope*tim
pevent<-exp(logit)/(1+exp(logit))
logpdf<-rep(0,length(tim))
P.obs = (y==0)*(1-pevent)+(y==1)*pevent
logpdf <- log(P.obs)
return(logpdf)
}
simulBinary<-function(psi,id,xidep) {
  tim<-xidep[,1]
  y<-xidep[,2]
  inter<-psi[id,1]
  slope<-psi[id,2]
  logit<-inter+slope*tim
  pevent<-1/(1+exp(-logit))
  ysim<-rbinom(length(tim),size=1, prob=pevent)
  return(ysim)
}
saemix.model<-saemixModel(model=binary.model,description="Binary model",simulate.function=simulBinary,
                           modeltype="likelihood",
                           psi0=matrix(c(0,-.5,0,0.5),ncol=2,byrow=TRUE,dimnames=list(NULL,c("theta1","theta2"),
                           transform.par=c(0,0), covariate.model=c(0,1),covariance.model=matrix(c(1,0,0,1),ncol=2,byrow=TRUE)))

##
##
## The following SaemixModel object was successfully created:
##
## Nonlinear mixed-effects model
##   Model function:  Binary model
##   Model type:     likelihood
## function(psi,id,xidep) {
##   tim<-xidep[,1]
##   y<-xidep[,2]
##   inter<-psi[id,1]
##   slope<-psi[id,2]
##   logit<-inter+slope*tim
##   pevent<-exp(logit)/(1+exp(logit))
##   logpdf<-rep(0,length(tim))
##   P.obs = (y==0)*(1-pevent)+(y==1)*pevent
##   logpdf <- log(P.obs)
##   return(logpdf)
## }
##   Nb of parameters: 2
##       parameter names:  theta1 theta2
##       distribution:
##       Parameter Distribution Estimated
## [1,] theta1      normal      Estimated
## [2,] theta2      normal      Estimated
##   Variance-covariance matrix:
##       theta1 theta2
## theta1      1      0

```

```

## theta2      0      1
## Covariate model:
##      theta1 theta2
## [1,]      0      1
##      Initial values
##      theta1 theta2
## Pop.CondInit      0    -0.5
## Cov.CondInit      0     0.5

saemix.options<-list(seed=1234567,save=FALSE,save.graphs=FALSE, displayProgress=FALSE, nb.chains=10, fi

# saemix fit
binary.fit<-saemix(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 toenail.saemix
##      Structured data: y ~ time + y | id
##      X variable for graphs: time ()
##      covariates: treatment (-)
##      reference class for covariate treatment : 0
## Dataset characteristics:
##      number of subjects:      294
##      number of observations: 1908
##      average/min/max nb obs: 6.49 / 1 / 7
## First 10 lines of data:
##      id      time y y.1 treatment mdv cens occ ytype
## 1  1  0.0000000 1  1      1  0  0  1  1
## 2  1  0.8571429 1  1      1  0  0  1  1
## 3  1  3.5357143 1  1      1  0  0  1  1
## 4  1  4.5357143 0  0      1  0  0  1  1
## 5  1  7.5357143 0  0      1  0  0  1  1
## 6  1 10.0357143 0  0      1  0  0  1  1
## 7  1 13.0714286 0  0      1  0  0  1  1
## 8  2  0.0000000 0  0      0  0  0  1  1
## 9  2  0.9642857 0  0      0  0  0  1  1
## 10 2  2.0000000 1  1      0  0  0  1  1
## -----
## ----          Model          ----
## -----
## Nonlinear mixed-effects model
##      Model function: Binary model
##      Model type: likelihood
## function(psi,id,xidep) {
##      tim<-xidep[,1]
##      y<-xidep[,2]
##      inter<-psi[id,1]
##      slope<-psi[id,2]
##      logit<-inter+slope*tim
##      pevent<-exp(logit)/(1+exp(logit))
##      logpdf<-rep(0,length(tim))

```

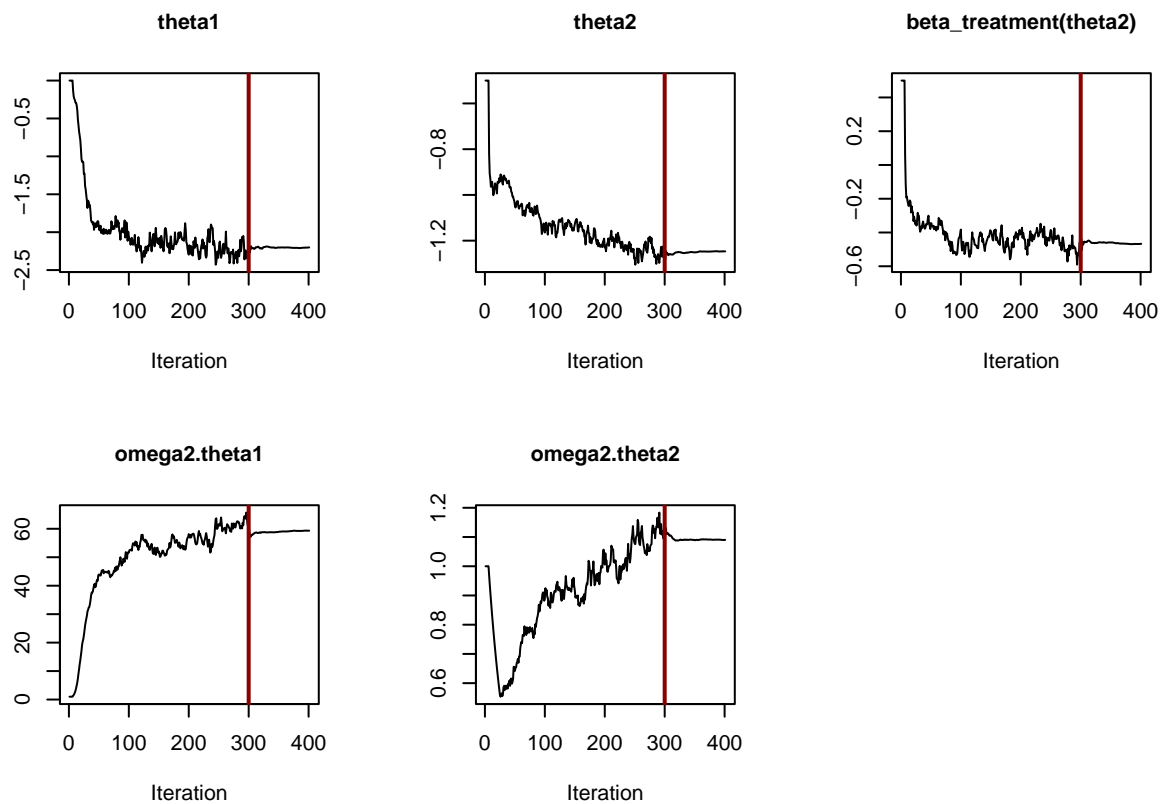
```

## P.obs = (y==0)*(1-pevent)+(y==1)*pevent
## logpdf <- log(P.obs)
## return(logpdf)
## }
## <bytecode: 0x55c274884f28>
## Nb of parameters: 2
## parameter names: theta1 theta2
## distribution:
## Parameter Distribution Estimated
## [1,] theta1 normal Estimated
## [2,] theta2 normal Estimated
## Variance-covariance matrix:
## theta1 theta2
## theta1 1 0
## theta2 0 1
## Covariate model:
## [,1] [,2]
## treatment 0 1
## Initial values
## theta1 theta2
## Pop.CondInit 0 -0.5
## Cov.CondInit 0 0.5
## -----
## ---- Key algorithm options ----
## -----
## Estimation of individual parameters (MAP)
## Estimation of log-likelihood by importance sampling
## Number of iterations: K1=300, K2=100
## Number of chains: 10
## Seed: 1234567
## 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
## [1,] theta1 -2.20
## [2,] theta2 -1.25
## [3,] beta_treatment(theta2) -0.47
## -----
## ----- Variance of random effects -----
## -----
## Parameter Estimate
## theta1 omega2.theta1 59.3
## theta2 omega2.theta2 1.1
## -----
## ----- Correlation matrix of random effects -----

```

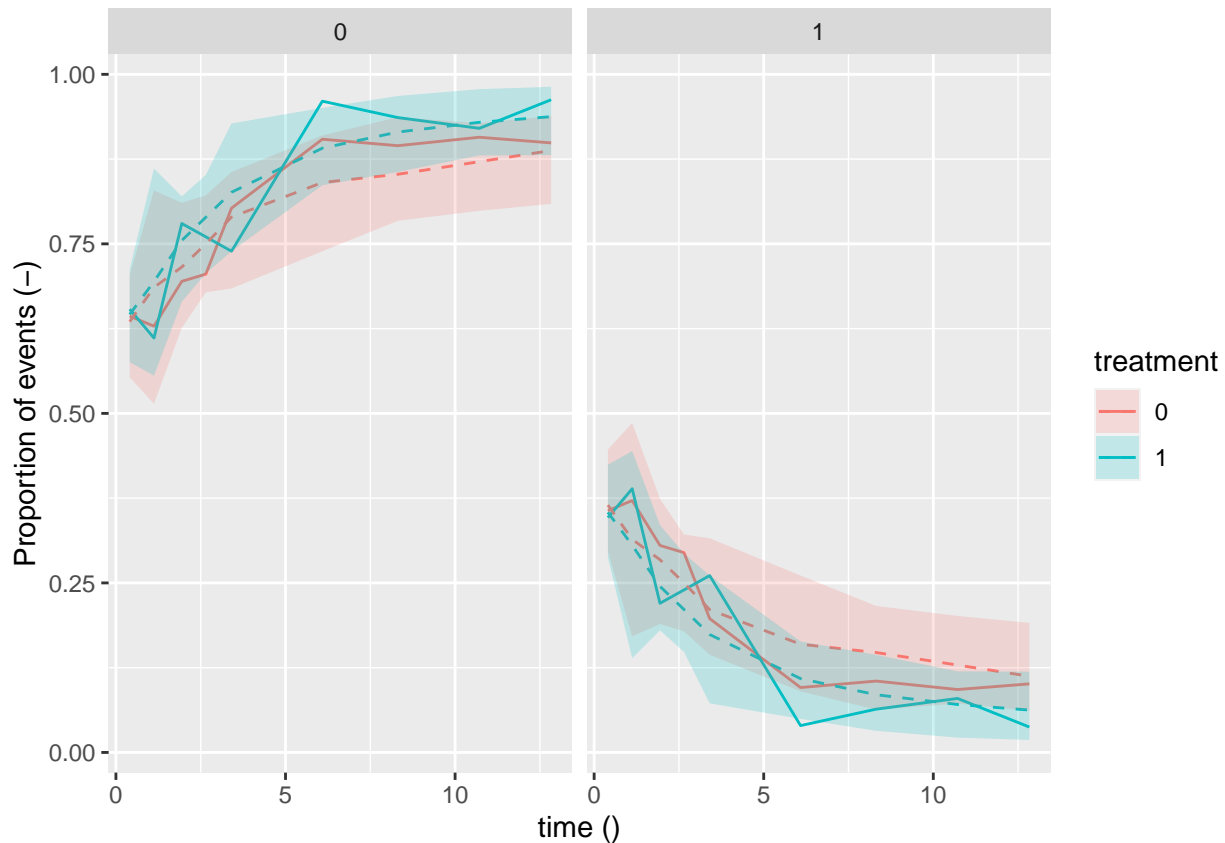
```
## -----
##               omega2.theta1 omega2.theta2
## omega2.theta1 1             0
## omega2.theta2 0             1
## -----
## ----- Statistical criteria -----
## -----
##
## Likelihood computed by importance sampling
##      -2LL= 1116.755
##      AIC = 1128.755
##      BIC = 1150.856
## -----
```

```
plot(binary.fit, plot.type="convergence")
```



```
# simulate from model (nsim=100)
yfit<-binary.fit
nsim<-100
yfit <- simulateDiscreteSaemix(yfit, nsim=nsim)

# VPC with function from library
discreteVPC(yfit, outcome="binary", which.cov="treatment")
```



```
# VPC with ggplot
simdat <- yfit@sim.data@datasim
simdat$visit <- rep(toenail.saemix$visit, nsim)
simdat$treatment <- rep(toenail.saemix$treatment, nsim)

# VPC-type diagnostic
ytab <- NULL
for(i in 1:nsim) {
  xtab <- simdat[simdat$irep == i, ]
  suppressMessages(
    xtab1 <- xtab %>%
      group_by(visit, treatment) %>%
      summarise(nev = sum(ysim), n = n()) %>%
      mutate(freq = nev/n)
  )
  ytab <- rbind(ytab, xtab1[, c("visit", "freq", "treatment")])
}
gtab <- ytab %>%
  group_by(visit, treatment) %>%
  summarise(lower = quantile(freq, c(0.05)), median = quantile(freq, c(0.5)), upper = quantile(freq, c(0.95)))
  mutate(treatment = ifelse(treatment == 1, "B", "A"))

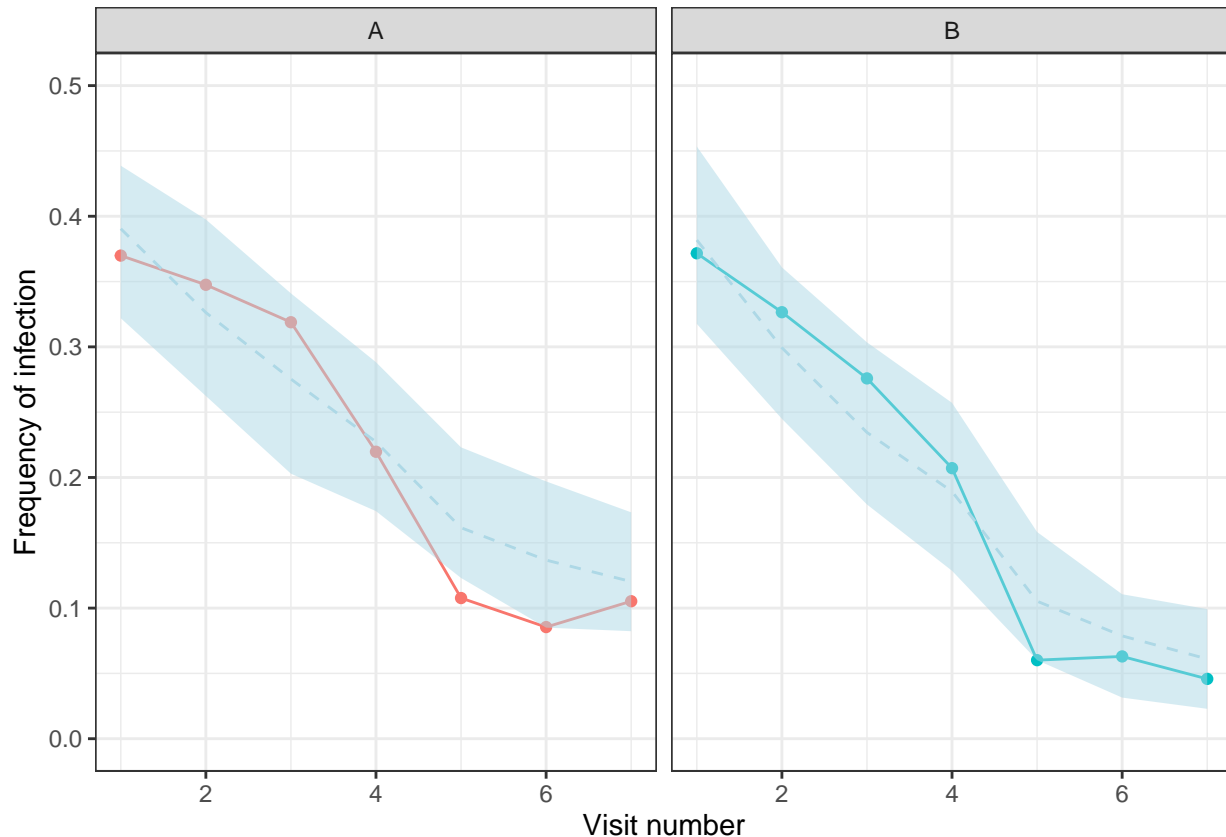
## `summarise()` has grouped output by 'visit'. You can override using the
## `.groups` argument.

gtab$freq <- 1

plot2 <- ggplot(toe1, aes(x=visit, y=freq, group=treatment)) + geom_line(aes(colour=treatment)) +
```

```
geom_point(aes(colour=treatment)) +
geom_line(data=gtab, aes(x=visit, y=median), linetype=2, colour='lightblue') +
geom_ribbon(data=gtab, aes(ymin=lower, ymax=upper), alpha=0.5, fill='lightblue') +
ylim(c(0,0.5)) + theme_bw() + theme(legend.position = "none") + facet_wrap(~treatment) +
xlab("Visit number") + ylab("Frequency of infection")
```

```
print(plot2)
```



```
if(saveForDocs) {
  namfig<-"toenail_barplotData.eps"
  cairo_ps(file = file.path(figDir, namfig), onefile = TRUE, fallback_resolution = 600, height=8.27, width=11.7,
    plot(plot2)
  dev.off()
}
```

Categorical response model

- Knee pain after 3, 7 and 10 days of treatment compared to baseline (time=0)
 - longitudinal ordinal model with 5 categories
 - similar results to Monolix in terms of parameter estimates
 - SE don't seem so off, but still higher than Monolix (but computed with linearisation anyway)
- Comparing the 3 covariate models - model with Age on alp1 and treatment on beta best

```
if(testMode)
  data(knee.saemix) else
  knee.saemix<-read.table(file.path(datDir, "knee.saemix.tab"), header=TRUE)
```

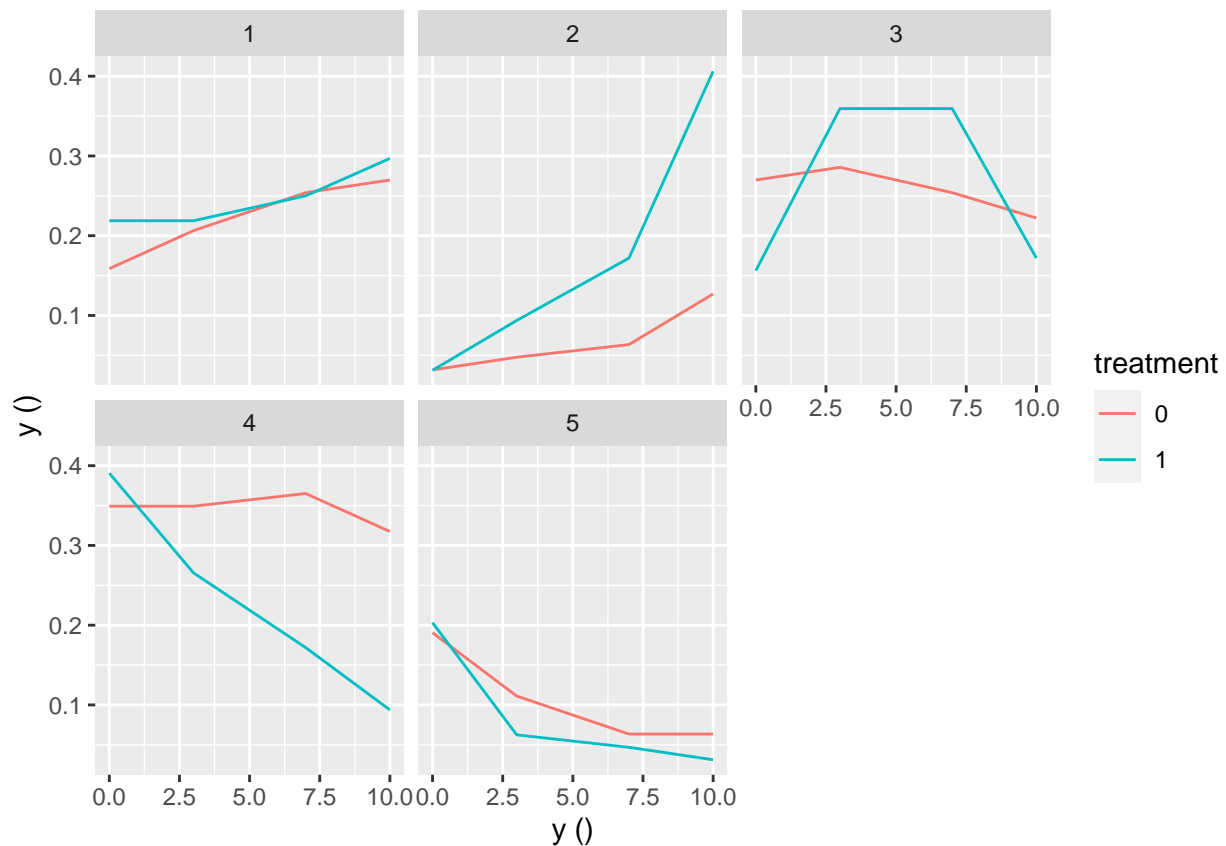


```
# Data
saemix.data<-saemixData(name.data=knee.saemix,name.group=c("id"),
                        name.predictors=c("y", "time"), name.X=c("time"),
                        name.covariates = c("Age","Sex","treatment","Age2"),
                        units=list(x="d",y="", covariates=c("yr","-","-", "yr2")))
```

```
## Column name(s) do(es) not exist in the dataset, please check
## Remove columns 1 ( )
## No valid name given, attempting automatic recognition
## Automatic recognition of columns y successful
##
##
## The following SaemixData object was successfully created:
##
## Object of class SaemixData
##   longitudinal data for use with the SAEM algorithm
## Dataset knee.saemix
##   Structured data: y ~ y + time | id
##   X variable for graphs: time (d)
##   covariates: Age (yr), Sex (-), treatment (-), Age2 (yr2)
##   reference class for covariate Sex : 0
##   reference class for covariate treatment : 0
```

```
# Plot of data with library
```

```
plotDiscreteData(saemix.data, outcome="categorical", which.cov="treatment")
```

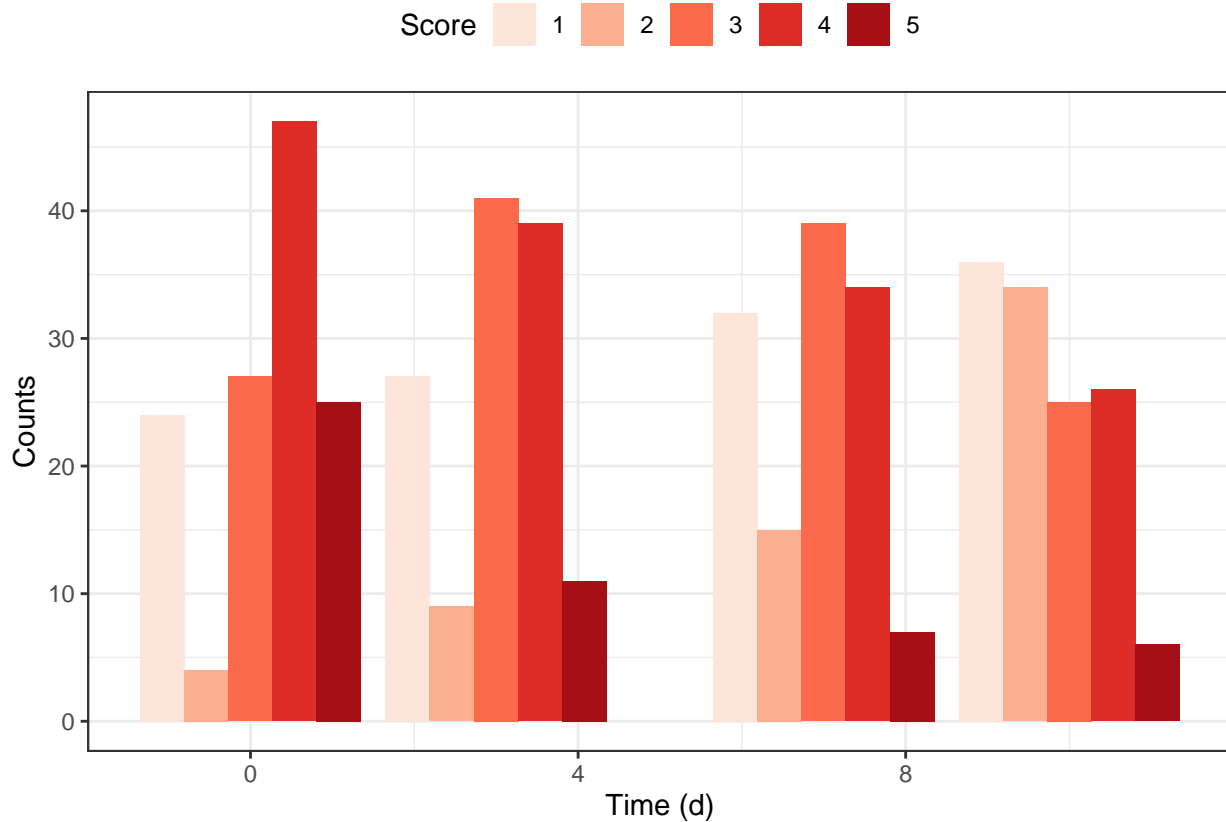


```
gtab <- knee.saemix %>%
```

```
group_by(time, y) %>%
  summarise(n=length(y)) %>%
  mutate(y=as.factor(y))
```

`summarise()` has grouped output by 'time'. You can override using the
`.groups` argument.

```
ggplot(data = gtab, aes(x = time, y=n, group=y, fill=y)) +
  geom_bar(stat="identity", position = "dodge") + theme_bw() +
  scale_fill_brewer(palette = "Reds") + theme(legend.position = "top") +
  labs(fill = "Score" + xlab("Time (d)") + ylab("Counts"))
```



```
# Model for ordinal responses
ordinal.model<-function(psi,id,xidep) {
  y<-xidep[,1]
  time<-xidep[,2]
  alp1<-psi[id,1]
  alp2<-psi[id,2]
  alp3<-psi[id,3]
  alp4<-psi[id,4]
  beta<-psi[id,5]

  logit1<-alp1 + beta*time
  logit2<-logit1+alp2
  logit3<-logit2+alp3
  logit4<-logit3+alp4
  pge1<-exp(logit1)/(1+exp(logit1))
  pge2<-exp(logit2)/(1+exp(logit2))
```

```

pge3<-exp(logit3)/(1+exp(logit3))
pge4<-exp(logit4)/(1+exp(logit4))
pobs = (y==1)*pge1+(y==2)*(pge2 - pge1)+(y==3)*(pge3 - pge2)+(y==4)*(pge4 - pge3)+(y==5)*(1 - pge4)
logpdf <- log(pobs)

return(logpdf)
}
simulateOrdinal<-function(psi,id,xidep) {
  y<-xidep[,1]
  time<-xidep[,2]
  alp1<-psi[id,1]
  alp2<-psi[id,2]
  alp3<-psi[id,3]
  alp4<-psi[id,4]
  beta<-psi[id,5]

  logit1<-alp1 + beta*time
  logit2<-logit1+alp2
  logit3<-logit2+alp3
  logit4<-logit3+alp4
  pge1<-exp(logit1)/(1+exp(logit1))
  pge2<-exp(logit2)/(1+exp(logit2))
  pge3<-exp(logit3)/(1+exp(logit3))
  pge4<-exp(logit4)/(1+exp(logit4))
  x<-runif(length(time))
  ysim<-1+as.integer(x>pge1)+as.integer(x>pge2)+as.integer(x>pge3)+as.integer(x>pge4)
  return(ysim)
}

# Fitting
covmodel2<-covmodel1<-matrix(data=0,ncol=5,nrow=4)
covmodel1[,1]<-1
covmodel1[,5]<-1
covmodel2[3,5]<-covmodel2[4,1]<-1

saemix.model<-saemixModel(model=ordinal.model,description="Ordinal categorical model",modeltype="likelihood",
  psi0=matrix(c(0,0.2, 0.6, 3, 0.2),ncol=5,byrow=TRUE,dimnames=list(NULL,c("alp1","alp2","alp3","alp4","beta")),
  transform.par=c(0,1,1,1,1),omega.init=diag(rep(1,5)), covariance.model = diag(5,1))

##
##
## The following SaemixModel object was successfully created:
##
## Nonlinear mixed-effects model
## Model function: Ordinal categorical model
## Model type: likelihood
## function(psi,id,xidep) {
##   y<-xidep[,1]
##   time<-xidep[,2]
##   alp1<-psi[id,1]
##   alp2<-psi[id,2]
##   alp3<-psi[id,3]
##   alp4<-psi[id,4]
##   beta<-psi[id,5]

```

```

##
## logit1<-alp1 + beta*time
## logit2<-logit1+alp2
## logit3<-logit2+alp3
## logit4<-logit3+alp4
## pge1<-exp(logit1)/(1+exp(logit1))
## pge2<-exp(logit2)/(1+exp(logit2))
## pge3<-exp(logit3)/(1+exp(logit3))
## pge4<-exp(logit4)/(1+exp(logit4))
## pobs = (y==1)*pge1+(y==2)*(pge2 - pge1)+(y==3)*(pge3 - pge2)+(y==4)*(pge4 - pge3)+(y==5)*(1 - pge4)
## logpdf <- log(pobs)
##
## return(logpdf)
## }
## Nb of parameters: 5
## parameter names: alp1 alp2 alp3 alp4 beta
## distribution:
## Parameter Distribution Estimated
## [1,] alp1 normal Estimated
## [2,] alp2 log-normal Estimated
## [3,] alp3 log-normal Estimated
## [4,] alp4 log-normal Estimated
## [5,] beta log-normal Estimated
## Variance-covariance matrix:
## alp1 alp2 alp3 alp4 beta
## alp1 1 0 0 0 0
## alp2 0 0 0 0 0
## alp3 0 0 0 0 0
## alp4 0 0 0 0 0
## beta 0 0 0 0 1
## No covariate in the model.
## Initial values
## alp1 alp2 alp3 alp4 beta
## Pop.CondInit 0 0.2 0.6 3 0.2

saemix.model.cov1<-saemixModel(model=ordinal.model,description="Ordinal categorical model",modeltype="1",
                                simulate.function=simulateOrdinal,
                                psi0=matrix(c(0,0.2, 0.6, 3, 0.2),ncol=5,byrow=TRUE,dimnames=list(NULL,c(
                                transform.par=c(0,1,1,1,1),omega.init=diag(rep(1,5)), covariance.model = c
                                covariate.model = covmodel1))

##
##
## The following SaemixModel object was successfully created:
##
## Nonlinear mixed-effects model
## Model function: Ordinal categorical model
## Model type: likelihood
## function(psi,id,xidep) {
## y<-xidep[,1]
## time<-xidep[,2]
## alp1<-psi[id,1]
## alp2<-psi[id,2]
## alp3<-psi[id,3]
## alp4<-psi[id,4]

```

```

##   beta<-psi[id,5]
##
##   logit1<-alp1 + beta*time
##   logit2<-logit1+alp2
##   logit3<-logit2+alp3
##   logit4<-logit3+alp4
##   pge1<-exp(logit1)/(1+exp(logit1))
##   pge2<-exp(logit2)/(1+exp(logit2))
##   pge3<-exp(logit3)/(1+exp(logit3))
##   pge4<-exp(logit4)/(1+exp(logit4))
##   pobs = (y==1)*pge1+(y==2)*(pge2 - pge1)+(y==3)*(pge3 - pge2)+(y==4)*(pge4 - pge3)+(y==5)*(1 - pge4)
##   logpdf <- log(pobs)
##
##   return(logpdf)
## }
##   Nb of parameters: 5
##       parameter names:  alp1 alp2 alp3 alp4 beta
##       distribution:
##       Parameter Distribution Estimated
## [1,] alp1      normal      Estimated
## [2,] alp2      log-normal   Estimated
## [3,] alp3      log-normal   Estimated
## [4,] alp4      log-normal   Estimated
## [5,] beta      log-normal   Estimated
##   Variance-covariance matrix:
##       alp1 alp2 alp3 alp4 beta
## alp1    1    0    0    0    0
## alp2    0    0    0    0    0
## alp3    0    0    0    0    0
## alp4    0    0    0    0    0
## beta    0    0    0    0    1
##   Covariate model:
##       alp1 alp2 alp3 alp4 beta
## [1,]    1    0    0    0    1
## [2,]    1    0    0    0    1
## [3,]    1    0    0    0    1
## [4,]    1    0    0    0    1
##   Initial values
##       alp1 alp2 alp3 alp4 beta
## Pop.CondInit    0 0.2 0.6   3 0.2
## Cov.CondInit    0 0.0 0.0   0 0.0
saemix.model.cov2<-saemixModel(model=ordinal.model,description="Ordinal categorical model",modeltype="1",
                                simulate.function=simulateOrdinal,
                                psi0=matrix(c(0,0.2, 0.6, 3, 0.2),ncol=5,byrow=TRUE,dimnames=list(NULL,c
                                transform.par=c(0,1,1,1,1),omega.init=diag(rep(1,5)), covariance.model =
                                covariate.model = covmodel2)

##
##
## The following SaemixModel object was successfully created:
##
## Nonlinear mixed-effects model
##   Model function:  Ordinal categorical model
##   Model type:     likelihood

```

```

## function(psi,id,xidep) {
##   y<-xidep[,1]
##   time<-xidep[,2]
##   alp1<-psi[id,1]
##   alp2<-psi[id,2]
##   alp3<-psi[id,3]
##   alp4<-psi[id,4]
##   beta<-psi[id,5]
##
##   logit1<-alp1 + beta*time
##   logit2<-logit1+alp2
##   logit3<-logit2+alp3
##   logit4<-logit3+alp4
##   pge1<-exp(logit1)/(1+exp(logit1))
##   pge2<-exp(logit2)/(1+exp(logit2))
##   pge3<-exp(logit3)/(1+exp(logit3))
##   pge4<-exp(logit4)/(1+exp(logit4))
##   pobs = (y==1)*pge1+(y==2)*(pge2 - pge1)+(y==3)*(pge3 - pge2)+(y==4)*(pge4 - pge3)+(y==5)*(1 - pge4)
##   logpdf <- log(pobs)
##
##   return(logpdf)
## }
##
##   Nb of parameters: 5
##       parameter names:  alp1 alp2 alp3 alp4 beta
##       distribution:
##       Parameter Distribution Estimated
## [1,] alp1      normal      Estimated
## [2,] alp2      log-normal   Estimated
## [3,] alp3      log-normal   Estimated
## [4,] alp4      log-normal   Estimated
## [5,] beta      log-normal   Estimated
##   Variance-covariance matrix:
##       alp1 alp2 alp3 alp4 beta
## alp1      1    0    0    0    0
## alp2      0    0    0    0    0
## alp3      0    0    0    0    0
## alp4      0    0    0    0    0
## beta      0    0    0    0    1
##   Covariate model:
##       alp1 alp2 alp3 alp4 beta
## [1,]      0    0    0    0    0
## [2,]      0    0    0    0    0
## [3,]      0    0    0    0    1
## [4,]      1    0    0    0    0
##   Initial values
##       alp1 alp2 alp3 alp4 beta
## Pop.CondInit      0 0.2 0.6   3 0.2
## Cov.CondInit      0 0.0 0.0   0 0.0

saemix.options<-list(seed=632545,save=FALSE,save.graphs=FALSE, nb.chains=10)
#saemix.options<-list(seed=632545,save=FALSE,save.graphs=FALSE, nb.chains=10, fim=FALSE)

ord.fit<-saemix(saemix.model,saemix.data,saemix.options)

## Error in solve.default(F0) :

```

```

## Routine Lapack dgesv : le système est exactement singulier : U[2,2] = 0
## Nonlinear mixed-effects model fit by the SAEM algorithm
## -----
## ----          Data          ----
## -----
## Object of class SaemixData
## longitudinal data for use with the SAEM algorithm
## Dataset knee.saemix
## Structured data: y ~ y + time | id
## X variable for graphs: time (d)
## covariates: Age (yr), Sex (-), treatment (-), Age2 (yr2)
## reference class for covariate Sex : 0
## reference class for covariate treatment : 0
## Dataset characteristics:
## number of subjects: 127
## number of observations: 508
## average/min/max nb obs: 4.00 / 4 / 4
## First 10 lines of data:
## id y time y.1 Age Sex treatment Age2 mdv cens occ ytype
## 1 1 4 0 4 -2 1 0 4 0 0 1 1
## 2 1 4 3 4 -2 1 0 4 0 0 1 1
## 3 1 4 7 4 -2 1 0 4 0 0 1 1
## 4 1 4 10 4 -2 1 0 4 0 0 1 1
## 5 2 4 0 4 2 1 0 4 0 0 1 1
## 6 2 4 3 4 2 1 0 4 0 0 1 1
## 7 2 4 7 4 2 1 0 4 0 0 1 1
## 8 2 4 10 4 2 1 0 4 0 0 1 1
## 9 3 3 0 3 11 1 0 121 0 0 1 1
## 10 3 3 3 3 11 1 0 121 0 0 1 1
## -----
## ----          Model          ----
## -----
## Nonlinear mixed-effects model
## Model function: Ordinal categorical model
## Model type: likelihood
## function(psi,id,xidep) {
## y<-xidep[,1]
## time<-xidep[,2]
## alp1<-psi[id,1]
## alp2<-psi[id,2]
## alp3<-psi[id,3]
## alp4<-psi[id,4]
## beta<-psi[id,5]
##
## logit1<-alp1 + beta*time
## logit2<-logit1+alp2
## logit3<-logit2+alp3
## logit4<-logit3+alp4
## pge1<-exp(logit1)/(1+exp(logit1))
## pge2<-exp(logit2)/(1+exp(logit2))
## pge3<-exp(logit3)/(1+exp(logit3))
## pge4<-exp(logit4)/(1+exp(logit4))
## pobs = (y==1)*pge1+(y==2)*(pge2 - pge1)+(y==3)*(pge3 - pge2)+(y==4)*(pge4 - pge3)+(y==5)*(1 - pge4)
## logpdf <- log(pobs)

```

```

##
##   return(logpdf)
## }
## <bytecode: 0x55c26f949db0>
##   Nb of parameters: 5
##     parameter names:  alp1 alp2 alp3 alp4 beta
##     distribution:
##       Parameter Distribution Estimated
## [1,] alp1      normal      Estimated
## [2,] alp2      log-normal   Estimated
## [3,] alp3      log-normal   Estimated
## [4,] alp4      log-normal   Estimated
## [5,] beta      log-normal   Estimated
##   Variance-covariance matrix:
##     alp1 alp2 alp3 alp4 beta
## alp1    1    0    0    0    0
## alp2    0    0    0    0    0
## alp3    0    0    0    0    0
## alp4    0    0    0    0    0
## beta    0    0    0    0    1
##   No covariate in the model.
##   Initial values
##           alp1 alp2 alp3 alp4 beta
## Pop.CondInit    0 0.2 0.6   3 0.2
## -----
## ----   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:  10
##   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,] alp1      -12.47  1.96 16
## [2,] alp2       5.34  1.93 36
## [3,] alp3       7.05  1.56 22
## [4,] alp4      10.31  3.03 29
## [5,] beta       0.71  0.15 22
## -----
## ----- Variance of random effects -----
## -----

```



```

##      Parameter   Estimate SE CV(%)
## alp1 omega2.alp1 129.61   NA NA
## beta omega2.beta   0.51   NA NA
## -----
## ----- Correlation matrix of random effects -----
## -----
##              omega2.alp1 omega2.beta
## omega2.alp1 1              0
## omega2.beta 0              1
## -----
## ----- Statistical criteria -----
## -----
## Likelihood computed by linearisation
##      -2LL= 5970.576
##      AIC = 5986.576
##      BIC = 6009.33
##
## Likelihood computed by importance sampling
##      -2LL= 864.4609
##      AIC = 880.4609
##      BIC = 903.2144
## -----
ord.fit.cov1<-saemix(saemix.model.cov1,saemix.data,saemix.options)

## Error in solve.default(F0) :
## Routine Lapack dgesv : le système est exactement singulier : U[2,2] = 0
## Nonlinear mixed-effects model fit by the SAEM algorithm
## -----
## ---- Data ----
## -----
## Object of class SaemixData
## longitudinal data for use with the SAEM algorithm
## Dataset knee.saemix
## Structured data: y ~ y + time | id
## X variable for graphs: time (d)
## covariates: Age (yr), Sex (-), treatment (-), Age2 (yr2)
## reference class for covariate Sex : 0
## reference class for covariate treatment : 0
## Dataset characteristics:
## number of subjects: 127
## number of observations: 508
## average/min/max nb obs: 4.00 / 4 / 4
## First 10 lines of data:
## id y time y.1 Age Sex treatment Age2 mdv cens occ ytype
## 1 1 4 0 4 -2 1 0 4 0 0 1 1
## 2 1 4 3 4 -2 1 0 4 0 0 1 1
## 3 1 4 7 4 -2 1 0 4 0 0 1 1
## 4 1 4 10 4 -2 1 0 4 0 0 1 1
## 5 2 4 0 4 2 1 0 4 0 0 1 1
## 6 2 4 3 4 2 1 0 4 0 0 1 1
## 7 2 4 7 4 2 1 0 4 0 0 1 1
## 8 2 4 10 4 2 1 0 4 0 0 1 1
## 9 3 3 0 3 11 1 0 121 0 0 1 1
## 10 3 3 3 3 11 1 0 121 0 0 1 1

```

```

## -----
## ----          Model          ----
## -----
## Nonlinear mixed-effects model
##   Model function: Ordinal categorical model
##   Model type: likelihood
## function(psi,id,xidep) {
##   y<-xidep[,1]
##   time<-xidep[,2]
##   alp1<-psi[id,1]
##   alp2<-psi[id,2]
##   alp3<-psi[id,3]
##   alp4<-psi[id,4]
##   beta<-psi[id,5]
##
##   logit1<-alp1 + beta*time
##   logit2<-logit1+alp2
##   logit3<-logit2+alp3
##   logit4<-logit3+alp4
##   pge1<-exp(logit1)/(1+exp(logit1))
##   pge2<-exp(logit2)/(1+exp(logit2))
##   pge3<-exp(logit3)/(1+exp(logit3))
##   pge4<-exp(logit4)/(1+exp(logit4))
##   pobs = (y==1)*pge1+(y==2)*(pge2 - pge1)+(y==3)*(pge3 - pge2)+(y==4)*(pge4 - pge3)+(y==5)*(1 - pge4)
##   logpdf <- log(pobs)
##
##   return(logpdf)
## }
## <bytecode: 0x55c26f949db0>
##   Nb of parameters: 5
##     parameter names: alp1 alp2 alp3 alp4 beta
##     distribution:
##     Parameter Distribution Estimated
## [1,] alp1      normal      Estimated
## [2,] alp2      log-normal   Estimated
## [3,] alp3      log-normal   Estimated
## [4,] alp4      log-normal   Estimated
## [5,] beta      log-normal   Estimated
##   Variance-covariance matrix:
##     alp1 alp2 alp3 alp4 beta
## alp1    1    0    0    0    0
## alp2    0    0    0    0    0
## alp3    0    0    0    0    0
## alp4    0    0    0    0    0
## beta    0    0    0    0    1
##   Covariate model:
##     [,1] [,2] [,3] [,4] [,5]
## Age      1    0    0    0    1
## Sex       1    0    0    0    1
## treatment 1    0    0    0    1
## Age2      1    0    0    0    1
##   Initial values
##     alp1 alp2 alp3 alp4 beta
## Pop.CondInit    0 0.2 0.6    3 0.2

```

```

## Cov.CondInit      0  0.0  0.0    0  0.0
## psi1              0  0.0  0.0    0  0.0
## psi1              0  0.0  0.0    0  0.0
## psi1              0  0.0  0.0    0  0.0
## -----
## ----      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:  10
##      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(%) p-value
## [1,] alp1           -1.6e+01 3.4778   22  -
## [2,] beta_Age(alp1)   1.2e-01 0.1171   95  0.292
## [3,] beta_Sex(alp1)  -6.9e-01 2.5495  370  0.787
## [4,] beta_treatment(alp1) 1.7e+00 2.1382  124  0.420
## [5,] beta_Age2(alp1)  3.4e-02 0.0166   49  0.041
## [6,] alp2           5.2e+00 1.8170   35  -
## [7,] alp3           6.9e+00 1.5416   22  -
## [8,] alp4           9.8e+00 2.8741   29  -
## [9,] beta           4.8e-01 0.2935   61  -
## [10,] beta_Age(beta) -1.6e-02 0.0229  142  0.480
## [11,] beta_Sex(beta)  3.4e-02 0.4957 1446  0.945
## [12,] beta_treatment(beta) 5.1e-01 0.4301   85  0.239
## [13,] beta_Age2(beta)  7.4e-04 0.0028  381  0.793
## -----
## -----      Variance of random effects      -----
## -----
##      Parameter      Estimate SE CV(%)
## alp1 omega2.alp1 108.43   NA NA
## beta omega2.beta   0.41   NA NA
## -----
## -----      Correlation matrix of random effects      -----
## -----
##      omega2.alp1 omega2.beta
## omega2.alp1 1      0
## omega2.beta 0      1
## -----
## -----      Statistical criteria      -----
## -----

```

```

## Likelihood computed by linearisation
##      -2LL= 5958.536
##      AIC = 5990.536
##      BIC = 6036.043
##
## Likelihood computed by importance sampling
##      -2LL= 840.4144
##      AIC = 872.4144
##      BIC = 917.9213
## -----
ord.fit.cov2<-saemix(saemix.model.cov2,saemix.data,saemix.options)

## Error in solve.default(F0) :
##   Routine Lapack dgesv : le système est exactement singulier : U[2,2] = 0
## Nonlinear mixed-effects model fit by the SAEM algorithm
## -----
## ----          Data          ----
## -----
## Object of class SaemixData
##   longitudinal data for use with the SAEM algorithm
## Dataset knee.saemix
##   Structured data: y ~ y + time | id
##   X variable for graphs: time (d)
##   covariates: Age (yr), Sex (-), treatment (-), Age2 (yr2)
##   reference class for covariate Sex : 0
##   reference class for covariate treatment : 0
## Dataset characteristics:
##   number of subjects:      127
##   number of observations: 508
##   average/min/max nb obs: 4.00 / 4 / 4
## First 10 lines of data:
##   id y time y.1 Age Sex treatment Age2 mdv cens occ ytype
## 1  1 4   0  4 -2  1      0  4  0  0  1  1
## 2  1 4   3  4 -2  1      0  4  0  0  1  1
## 3  1 4   7  4 -2  1      0  4  0  0  1  1
## 4  1 4  10  4 -2  1      0  4  0  0  1  1
## 5  2 4   0  4  2  1      0  4  0  0  1  1
## 6  2 4   3  4  2  1      0  4  0  0  1  1
## 7  2 4   7  4  2  1      0  4  0  0  1  1
## 8  2 4  10  4  2  1      0  4  0  0  1  1
## 9  3 3   0  3 11  1      0 121  0  0  1  1
## 10 3 3   3  3 11  1      0 121  0  0  1  1
## -----
## ----          Model          ----
## -----
## Nonlinear mixed-effects model
##   Model function: Ordinal categorical model
##   Model type: likelihood
## function(psi,id,xidep) {
##   y<-xidep[,1]
##   time<-xidep[,2]
##   alp1<-psi[id,1]
##   alp2<-psi[id,2]
##   alp3<-psi[id,3]

```

```

## alp4<-psi[id,4]
## beta<-psi[id,5]
##
## logit1<-alp1 + beta*time
## logit2<-logit1+alp2
## logit3<-logit2+alp3
## logit4<-logit3+alp4
## pge1<-exp(logit1)/(1+exp(logit1))
## pge2<-exp(logit2)/(1+exp(logit2))
## pge3<-exp(logit3)/(1+exp(logit3))
## pge4<-exp(logit4)/(1+exp(logit4))
## pobs = (y==1)*pge1+(y==2)*(pge2 - pge1)+(y==3)*(pge3 - pge2)+(y==4)*(pge4 - pge3)+(y==5)*(1 - pge4)
## logpdf <- log(pobs)
##
## return(logpdf)
## }
## <bytecode: 0x55c26f949db0>
## Nb of parameters: 5
## parameter names: alp1 alp2 alp3 alp4 beta
## distribution:
## Parameter Distribution Estimated
## [1,] alp1 normal Estimated
## [2,] alp2 log-normal Estimated
## [3,] alp3 log-normal Estimated
## [4,] alp4 log-normal Estimated
## [5,] beta log-normal Estimated
## Variance-covariance matrix:
## alp1 alp2 alp3 alp4 beta
## alp1 1 0 0 0 0
## alp2 0 0 0 0 0
## alp3 0 0 0 0 0
## alp4 0 0 0 0 0
## beta 0 0 0 0 1
## Covariate model:
## [,1] [,2] [,3] [,4] [,5]
## treatment 0 0 0 0 1
## Age2 1 0 0 0 0
## Initial values
## alp1 alp2 alp3 alp4 beta
## Pop.CondInit 0 0.2 0.6 3 0.2
## Cov.CondInit 0 0.0 0.0 0 0.0
## psi1 0 0.0 0.0 0 0.0
## -----
## ---- 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: 10
## 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(%) p-value
## [1,] alp1           -16.300 2.406 15      -
## [2,] beta_Age2(alp1)    0.041 0.014 33    0.0028
## [3,] alp2             5.340 1.814 34      -
## [4,] alp3             7.173 1.587 22      -
## [5,] alp4            10.079 3.010 30      -
## [6,] beta             0.535 0.176 33      -
## [7,] beta_treatment(beta) 0.554 0.347 63    0.1105
## -----
## ----- Variance of random effects -----
## -----
##      Parameter  Estimate SE CV(%)
## alp1 omega2.alp1 116.22  NA NA
## beta omega2.beta  0.45  NA NA
## -----
## ----- Correlation matrix of random effects -----
## -----
##      omega2.alp1 omega2.beta
## omega2.alp1 1      0
## omega2.beta 0      1
## -----
## ----- Statistical criteria -----
## -----
## Likelihood computed by linearisation
##      -2LL= 5980.122
##      AIC = 6000.122
##      BIC = 6028.564
##
## Likelihood computed by importance sampling
##      -2LL= 843.57
##      AIC = 863.57
##      BIC = 892.0119
## -----

```

```
BIC(ord.fit)
```

```
## [1] 903.2144
```

```
BIC(ord.fit.cov1)
```

```
## [1] 917.9213
```

```
BIC(ord.fit.cov2)
```

```
## [1] 892.0119
```

```

# Comparing the 3 covariate models - model with Age2 on alp1 and treatment on beta best
compare.saemix(ord.fit, ord.fit.cov1, ord.fit.cov2)

```

```
## Likelihoods calculated by importance sampling
```

```
##      AIC      BIC  BIC.cov
## 1 880.4609 903.2144 892.8407
## 2 872.4144 917.9213 907.5477
## 3 863.5700 892.0119 881.6382
```

```
#####
```

```
# But VPC not good
```

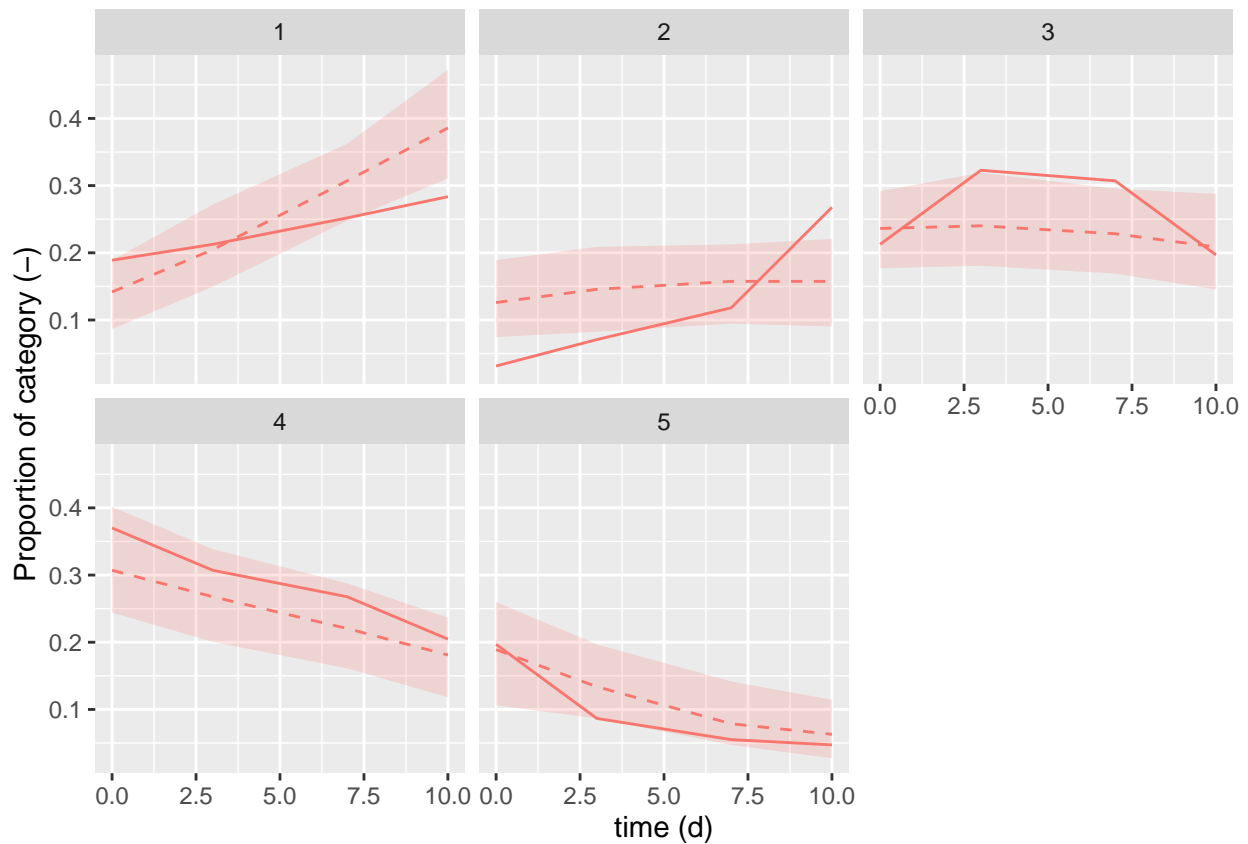
```
### Simulations for VPC
```

```
nsim<-100
```

```
yfit<-ord.fit.cov2
```

```
yfit<-simulateDiscreteSaemix(yfit, nsim=nsim)
```

```
discreteVPC(yfit, outcome="categorical")
```



```
# With ggplot2
```

```
simdat <-yfit@sim.data@datasim
```

```
simdat$time<-rep(yfit@data@data$time,nsim)
```

```
simdat$treatment<-rep(yfit@data@data$treatment,nsim)
```

```
ytab<-NULL
```

```
for(irep in 1:nsim) {
```

```
  xtab<-simdat[simdat$irep==irep,]
```

```
  suppressMessages(
```

```
    xtab1 <- xtab %>%
```

```
      group_by(time, treatment, ysim) %>%
```

```
      summarise(n=length(ysim))
```

```
  )
```

```

  ytab<-rbind(ytab,xtab1[,c("time","ysim","n","treatment")])
}
gtab <- ytab %>%
  group_by(time, treatment, ysim) %>%
  summarise(lower=quantile(n, c(0.05)), n=quantile(n, c(0.5)), upper=quantile(n, c(0.95))) %>%
  mutate(y=as.factor(ysim))

## `summarise()` has grouped output by 'time', 'treatment'. You can override using
## the `groups` argument.

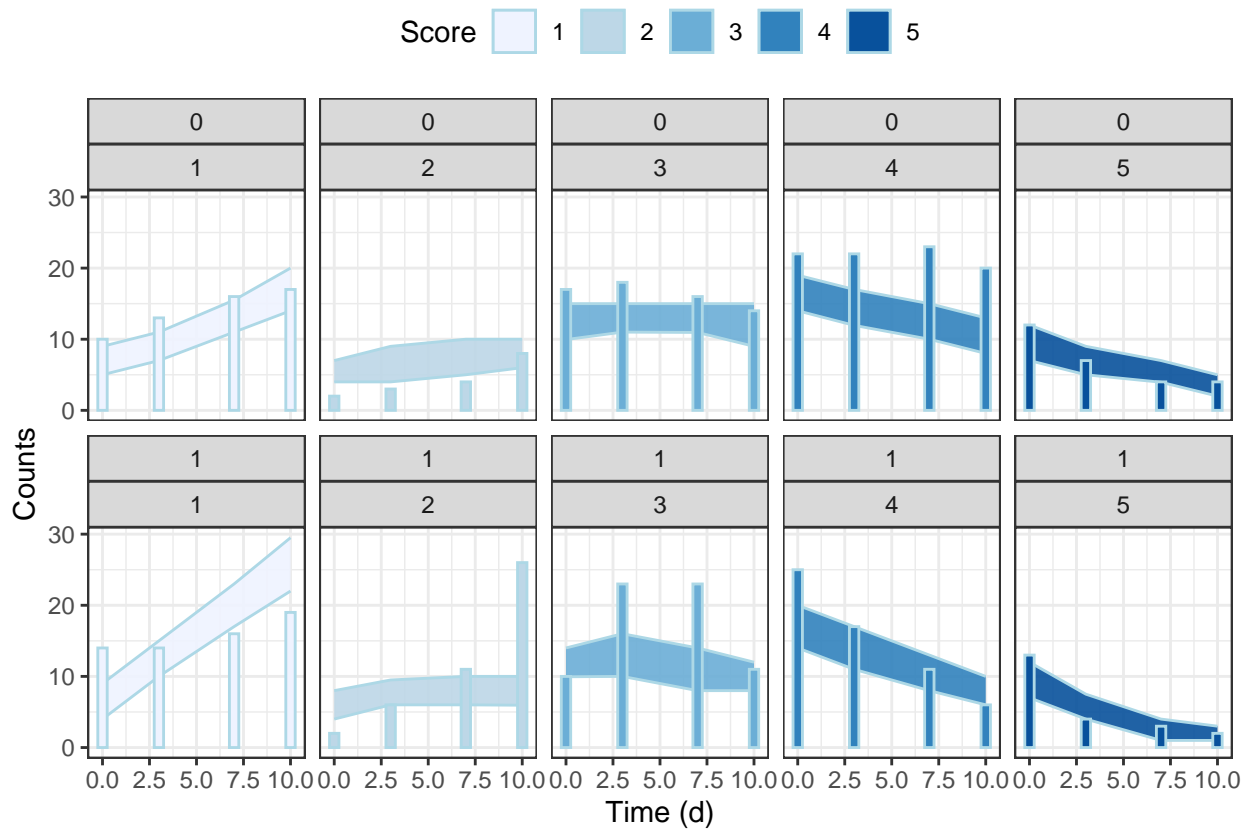
knee2 <- knee.saemix %>%
  group_by(time, treatment, y) %>%
  summarise(n=length(y)) %>%
  mutate(y=as.factor(y))

## `summarise()` has grouped output by 'time', 'treatment'. You can override using
## the `groups` argument.

kneevpc <- ggplot(data = knee2, aes(x = time, y=n, fill=y, group=treatment)) +
  geom_ribbon(data=gtab, aes(x=time, ymin=lower, ymax=upper), alpha=0.9, colour="lightblue") +
  geom_col(position = "dodge", width=0.5, colour="lightblue") + theme_bw() +
  scale_fill_brewer(palette = "Blues") + theme(legend.position = "top") +
  labs(fill = "Score") + xlab("Time (d)") + ylab("Counts") + facet_wrap(treatment~y, nrow=2)

print(kneevpc)

```



```

# VPC for median score in each group
knee3 <- knee.saemix %>%
  group_by(time, treatment) %>%

```



```

summarise(mean=mean(y))

## `summarise()` has grouped output by 'time'. You can override using the
## `.groups` argument.

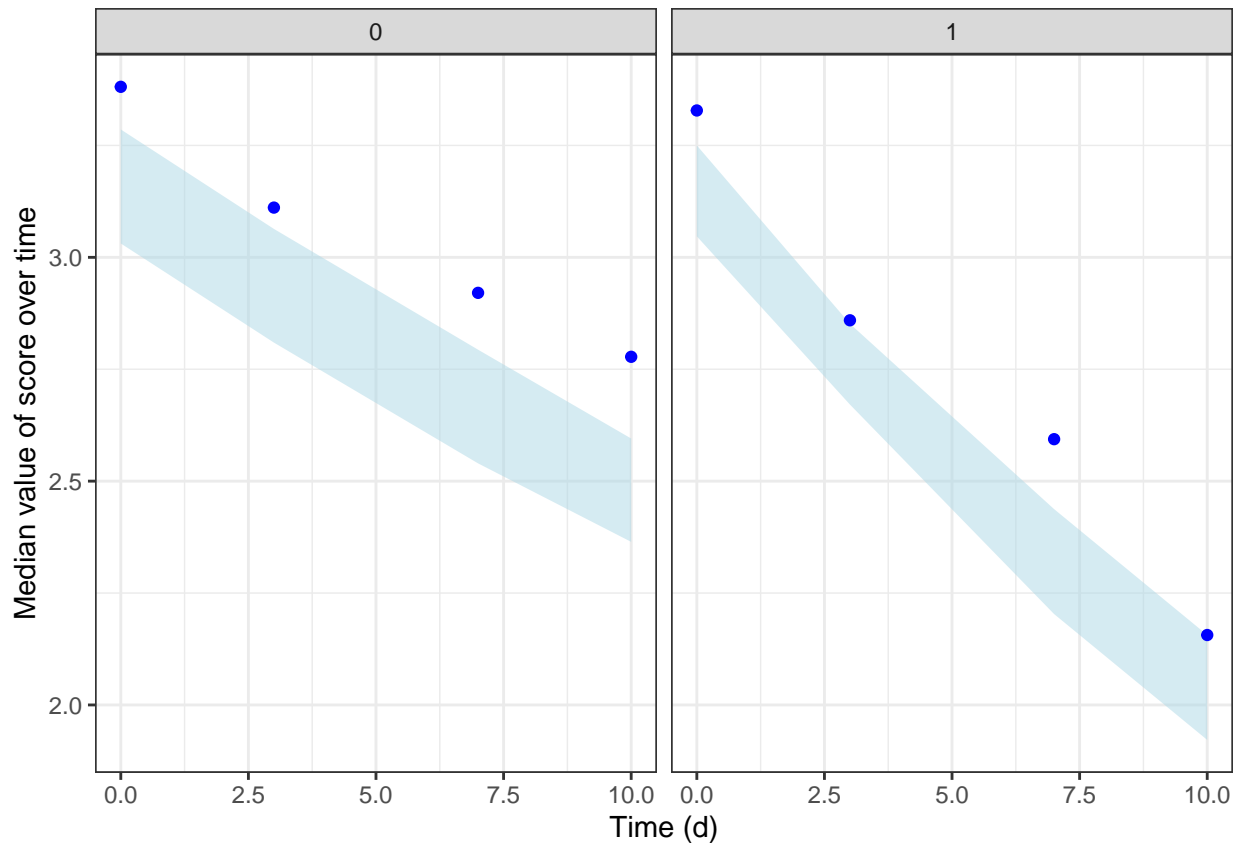
ytab<-NULL
for(irep in 1:nsim) {
  xtab<-simdat[simdat$irep==irep,]
  suppressMessages(
    xtab1 <- xtab %>%
      group_by(time, treatment) %>%
      summarise(mean=mean(ysim))
  )
  ytab<-rbind(ytab,xtab1[,c("time","treatment","mean")])
}
gtab <- ytab %>%
  group_by(time, treatment) %>%
  summarise(lower=quantile(mean, c(0.05)), mean=median(mean), upper=quantile(mean, c(0.95)))

## `summarise()` has grouped output by 'time'. You can override using the
## `.groups` argument.

kneeMedvpc <- ggplot(data = knee3, aes(x = time, y=mean, group=treatment)) +
  geom_ribbon(data=gtab, aes(x=time, ymin=lower, ymax=upper), alpha=0.5, fill="lightblue") +
  geom_point(colour='blue') + theme_bw() +
  scale_fill_brewer(palette = "Blues") + theme(legend.position = "top") +
  labs(fill = "Score") + xlab("Time (d)") + ylab("Median value of score over time") + facet_wrap(~trea

print(kneeMedvpc)

```



```
if(saveForDocs) {
  namfig<-"knee_medianScoreVPC.eps"
  cairo_ps(file = file.path(figDir, namfig), onefile = TRUE, fallback_resolution = 600, height=8.27, width=10.0)
  plot(kneeMedvpc)
  dev.off()
}
```

Count data model

- Epilepsy
 - dataset epil from MASS
 - very basic model with only one parameter
- Drinking patterns amongst students (David Atkins from tutorial)
 - dataset rapi.saemix
 - lambda parameter from Poisson model with a time-effect, gender effects on both intercept and slope
 - different models can be adjusted to the data, accounting for overdispersion

```
epilepsy<-MASS::epil
saemix.data<-saemixData(name.data=epilepsy, name.group=c("subject"),
  name.predictors=c("period","y"),name.response=c("y"),
  name.covariates=c("trt","base", "age"),
  units=list(x="2-week",y="",covariates=c("", "", "yr")))
```

Epilepsy data

```
##
```

```

##
## The following SaemixData object was successfully created:
##
## Object of class SaemixData
##      longitudinal data for use with the SAEM algorithm
## Dataset epilepsy
##      Structured data: y ~ period + y | subject
##      X variable for graphs: period (2-week)
##      covariates: trt (), base (), age (yr)
##      reference class for covariate trt : placebo

## Poisson model with one parameter
countmodel.poisson<-function(psi,id,xidep) {
  y<-xidep[,2]
  lambda<-psi[id,1]
  logp <- -lambda + y*log(lambda) - log(factorial(y))
  return(logp)
}

# Adding a period effect
countmodel.periodpoi<-function(psi,id,xidep) {
  tim <- xidep[,1]
  y<-xidep[,2]
  lam<-psi[id,1]
  betaT<-psi[id,2]
  lambda<-lam*exp(beta*log(tim))
  logp <- -lambda + y*log(lambda) - log(factorial(y))
  return(logp)
}

## Generalised Poisson model
countmodel.genpoisson<-function(psi,id,xidep) {
  y<-xidep[,1]
  delta<-psi[id,1]
  lambda<-psi[id,2]
  logp <- -lambda
  pos.ind <- which(y>0)
  lp1 <-log(lambda) + (y-1)*log(lambda+y*delta) - (lambda+y*delta) - log(factorial(y))
  logp[pos.ind] <- lp1[pos.ind]
  return(logp)
}

## Poisson model with Zero-Inflation
countmodel.zip<-function(psi,id,xidep) {
  y<-xidep[,2]
  lambda<-psi[id,1]
  p0<-psi[id,2]
  logp <- log(1-p0) -lambda + y*log(lambda) - log(factorial(y))
  logp0 <- log(p0+(1-p0)*exp(-lambda))
  logp[y==0]<-logp0[y==0]
  return(logp)
}

saemix.model.poi<-saemixModel(model=countmodel.poisson,description="count model Poisson",modeltype="lik

```

```

psi0=matrix(c(0.5),ncol=1,byrow=TRUE,dimnames=list(NULL, c("lambda"))),
transform.par=c(1))

##
##
## The following SaemixModel object was successfully created:
##
## Nonlinear mixed-effects model
##   Model function:  count model Poisson
##   Model type:  likelihood
## function(psi,id,xidep) {
##   y<-xidep[,2]
##   lambda<-psi[id,1]
##   logp <- -lambda + y*log(lambda) - log(factorial(y))
##   return(logp)
## }
##   Nb of parameters: 1
##     parameter names:  lambda
##     distribution:
##     Parameter Distribution Estimated
## [1,] lambda    log-normal    Estimated
##   Variance-covariance matrix:
##     lambda
## lambda      1
##   No covariate in the model.
##   Initial values
##     lambda
## Pop.CondInit    0.5
saemix.model.zip<-saemixModel(model=countmodel.zip,description="count model ZIP",modeltype="likelihood",
psi0=matrix(c(0.5,0.2),ncol=2,byrow=TRUE,dimnames=list(NULL, c("lambda","p0"))),
transform.par=c(1,3), #omega.init=matrix(c(0.5,0,0,0.3),ncol=2,byrow=TRUE),
covariance.model=matrix(c(1,0,0,0),ncol=2,byrow=TRUE))

##
##
## The following SaemixModel object was successfully created:
##
## Nonlinear mixed-effects model
##   Model function:  count model ZIP
##   Model type:  likelihood
## function(psi,id,xidep) {
##   y<-xidep[,2]
##   lambda<-psi[id,1]
##   p0<-psi[id,2]
##   logp <- log(1-p0) -lambda + y*log(lambda) - log(factorial(y))
##   logp0 <- log(p0+(1-p0)*exp(-lambda))
##   logp[y==0]<-logp0[y==0]
##   return(logp)
## }
##   Nb of parameters: 2
##     parameter names:  lambda p0
##     distribution:
##     Parameter Distribution Estimated

```

```

## [1,] lambda    log-normal    Estimated
## [2,] p0        logit          Estimated
##   Variance-covariance matrix:
##           lambda p0
## lambda      1    0
## p0           0    0
##   No covariate in the model.
##   Initial values
##           lambda p0
## Pop.CondInit    0.5 0.2

saemix.model.gp<-saemixModel(model=countmodel.zip,description="Generalised Poisson model",modeltype="li
                           psi0=matrix(c(0.5,0.2),ncol=2,byrow=TRUE,dimnames=list(NULL, c("delta","l
                           transform.par=c(1,1), #omega.init=matrix(c(0.5,0,0,0.3),ncol=2,byrow=TRUE,
                           covariance.model=matrix(c(1,0,0,0),ncol=2,byrow=TRUE))

##
##
## The following SaemixModel object was successfully created:
##
## Nonlinear mixed-effects model
##   Model function:  Generalised Poisson model
##   Model type:     likelihood
## function(psi,id,xidep) {
##   y<-xidep[,2]
##   lambda<-psi[id,1]
##   p0<-psi[id,2]
##   logp <- log(1-p0) -lambda + y*log(lambda) - log(factorial(y))
##   logp0 <- log(p0+(1-p0)*exp(-lambda))
##   logp[y==0]<-logp0[y==0]
##   return(logp)
## }
##   Nb of parameters: 2
##       parameter names:  delta lambda
##       distribution:
##       Parameter Distribution Estimated
## [1,] delta    log-normal    Estimated
## [2,] lambda   log-normal    Estimated
##   Variance-covariance matrix:
##           delta lambda
## delta      1      0
## lambda     0      0
##   No covariate in the model.
##   Initial values
##           delta lambda
## Pop.CondInit    0.5    0.2

saemix.options<-list(seed=632545,save=FALSE,save.graphs=FALSE, displayProgress=FALSE)

poisson.fit<-saemix(saemix.model.poi,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 epilepsy
##   Structured data: y ~ period + y | subject
##   X variable for graphs: period (2-week)
##   covariates: trt (), base (), age (yr)
##   reference class for covariate trt : placebo
## Dataset characteristics:
##   number of subjects:      59
##   number of observations: 236
##   average/min/max nb obs: 4.00 / 4 / 4
## First 10 lines of data:
##   subject period y y.1      trt base age mdv cens occ ytype
## 1         1      1 5    5 placebo 11 31  0   0   1    1
## 2         1      2 3    3 placebo 11 31  0   0   1    1
## 3         1      3 3    3 placebo 11 31  0   0   1    1
## 4         1      4 3    3 placebo 11 31  0   0   1    1
## 5         2      1 3    3 placebo 11 30  0   0   1    1
## 6         2      2 5    5 placebo 11 30  0   0   1    1
## 7         2      3 3    3 placebo 11 30  0   0   1    1
## 8         2      4 3    3 placebo 11 30  0   0   1    1
## 9         3      1 2    2 placebo  6 25  0   0   1    1
## 10        3      2 4    4 placebo  6 25  0   0   1    1
## -----
## ----          Model          ----
## -----
## Nonlinear mixed-effects model
##   Model function: count model Poisson
##   Model type: likelihood
## function(psi,id,xidep) {
##   y<-xidep[,2]
##   lambda<-psi[id,1]
##   logp <- -lambda + y*log(lambda) - log(factorial(y))
##   return(logp)
## }
## <bytecode: 0x55c27758ddd8>
##   Nb of parameters: 1
##     parameter names: lambda
##     distribution:
##       Parameter Distribution Estimated
## [1,] lambda    log-normal Estimated
##   Variance-covariance matrix:
##     lambda
## lambda      1
##   No covariate in the model.
##   Initial values
##     lambda
## Pop.CondInit    0.5
## -----
## ----   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,] lambda      5.1      0.71 14
## -----
## ----- Variance of random effects -----
## -----
##      Parameter      Estimate SE    CV(%)
## lambda omega2.lambda 0.9      0.21 23
## -----
## ----- Correlation matrix of random effects -----
## -----
##      omega2.lambda
## omega2.lambda 1
## -----
## ----- Statistical criteria -----
## -----
## Likelihood computed by linearisation
##      -2LL= 60096.92
##      AIC = 60102.92
##      BIC = 60109.15
##
## Likelihood computed by importance sampling
##      -2LL= 1402.095
##      AIC = 1408.095
##      BIC = 1414.327
## -----
genpoisson.fit<-saemix(saemix.model.gp,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 epilepsy
##      Structured data: y ~ period + y | subject
##      X variable for graphs: period (2-week)
##      covariates: trt (), base (), age (yr)
##      reference class for covariate trt :  placebo
## Dataset characteristics:

```

```

##      number of subjects:      59
##      number of observations: 236
##      average/min/max nb obs: 4.00 / 4 / 4
## First 10 lines of data:
##      subject period y y.1      trt base age mdv cens occ ytype
## 1          1      1 5    5 placebo 11 31  0   0  1    1
## 2          1      2 3    3 placebo 11 31  0   0  1    1
## 3          1      3 3    3 placebo 11 31  0   0  1    1
## 4          1      4 3    3 placebo 11 31  0   0  1    1
## 5          2      1 3    3 placebo 11 30  0   0  1    1
## 6          2      2 5    5 placebo 11 30  0   0  1    1
## 7          2      3 3    3 placebo 11 30  0   0  1    1
## 8          2      4 3    3 placebo 11 30  0   0  1    1
## 9          3      1 2    2 placebo  6 25  0   0  1    1
## 10         3      2 4    4 placebo  6 25  0   0  1    1
## -----
## ----          Model          ----
## -----
## Nonlinear mixed-effects model
## Model function: Generalised Poisson model
## Model type: likelihood
## function(psi,id,xidep) {
##   y<-xidep[,2]
##   lambda<-psi[id,1]
##   p0<-psi[id,2]
##   logp <- log(1-p0) -lambda + y*log(lambda) - log(factorial(y))
##   logp0 <- log(p0+(1-p0)*exp(-lambda))
##   logp[y==0]<-logp0[y==0]
##   return(logp)
## }
## <bytecode: 0x55c2784b48f0>
## Nb of parameters: 2
##   parameter names: delta lambda
##   distribution:
##   Parameter Distribution Estimated
## [1,] delta      log-normal Estimated
## [2,] lambda     log-normal Estimated
## Variance-covariance matrix:
##       delta lambda
## delta      1      0
## lambda     0      0
## No covariate in the model.
## Initial values
##       delta lambda
## Pop.CondInit  0.5   0.2
## -----
## ---- 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,] delta      5.314    0.747 14
## [2,] lambda     0.041    0.024 58
## -----
## ----- Variance of random effects -----
## -----
##      Parameter      Estimate SE      CV(%)
## delta omega2.delta 0.86      0.21 24
## -----
## ----- Correlation matrix of random effects -----
## -----
##      omega2.delta
## omega2.delta 1
## -----
## ----- Statistical criteria -----
## -----
## Likelihood computed by linearisation
##      -2LL= 60647.88
##      AIC = 60655.88
##      BIC = 60664.19
##
## Likelihood computed by importance sampling
##      -2LL= 1381.329
##      AIC = 1389.329
##      BIC = 1397.639
## -----
zippoisson.fit<-saemix(saemix.model.zip,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 epilepsy
##      Structured data: y ~ period + y | subject
##      X variable for graphs: period (2-week)
##      covariates: trt (), base (), age (yr)
##      reference class for covariate trt : placebo
## Dataset characteristics:
##      number of subjects:      59
##      number of observations: 236

```

```

##      average/min/max nb obs: 4.00 / 4 / 4
## First 10 lines of data:
##      subject period y y.1      trt base age mdv cens occ ytype
## 1          1      1 5    5 placebo 11 31  0  0  1    1
## 2          1      2 3    3 placebo 11 31  0  0  1    1
## 3          1      3 3    3 placebo 11 31  0  0  1    1
## 4          1      4 3    3 placebo 11 31  0  0  1    1
## 5          2      1 3    3 placebo 11 30  0  0  1    1
## 6          2      2 5    5 placebo 11 30  0  0  1    1
## 7          2      3 3    3 placebo 11 30  0  0  1    1
## 8          2      4 3    3 placebo 11 30  0  0  1    1
## 9          3      1 2    2 placebo  6 25  0  0  1    1
## 10         3      2 4    4 placebo  6 25  0  0  1    1
## -----
## ----          Model          ----
## -----
## Nonlinear mixed-effects model
## Model function: count model ZIP
## Model type: likelihood
## function(psi,id,xidep) {
##   y<-xidep[,2]
##   lambda<-psi[id,1]
##   p0<-psi[id,2]
##   logp <- log(1-p0) -lambda + y*log(lambda) - log(factorial(y))
##   logp0 <- log(p0+(1-p0)*exp(-lambda))
##   logp[y==0]<-logp0[y==0]
##   return(logp)
## }
## <bytecode: 0x55c2784b48f0>
## Nb of parameters: 2
##   parameter names: lambda p0
##   distribution:
##   Parameter Distribution Estimated
## [1,] lambda    log-normal    Estimated
## [2,] p0        logit          Estimated
## Variance-covariance matrix:
##      lambda p0
## lambda      1  0
## p0          0  0
## No covariate in the model.
## Initial values
##      lambda p0
## Pop.CondInit 0.5 0.2
## -----
## ---- 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,] lambda      5.320   0.748 14
## [2,] p0          0.041   0.024 58
## -----
## ----- Variance of random effects -----
## -----
##      Parameter      Estimate SE      CV(%)
## lambda omega2.lambda 0.86     0.21 24
## -----
## ----- Correlation matrix of random effects -----
## -----
##      omega2.lambda
## omega2.lambda 1
## -----
## ----- Statistical criteria -----
## -----
## Likelihood computed by linearisation
##      -2LL= 61045.94
##      AIC = 61053.94
##      BIC = 61062.25
##
## Likelihood computed by importance sampling
##      -2LL= 1381.314
##      AIC = 1389.314
##      BIC = 1397.624
## -----

```

```

if(testMode)
  data(rapi.saemix) else
  rapi.saemix<-read.table(file.path(datDir, "rapi.saemix.tab"), header=TRUE)

# Data
saemix.data<-saemixData(name.data=rapi.saemix, name.group=c("id"),
                        name.predictors=c("time", "rapi"), name.response=c("rapi"),
                        name.covariates=c("gender"),
                        units=list(x="months", y="", covariates=c("")))

```

RAPI

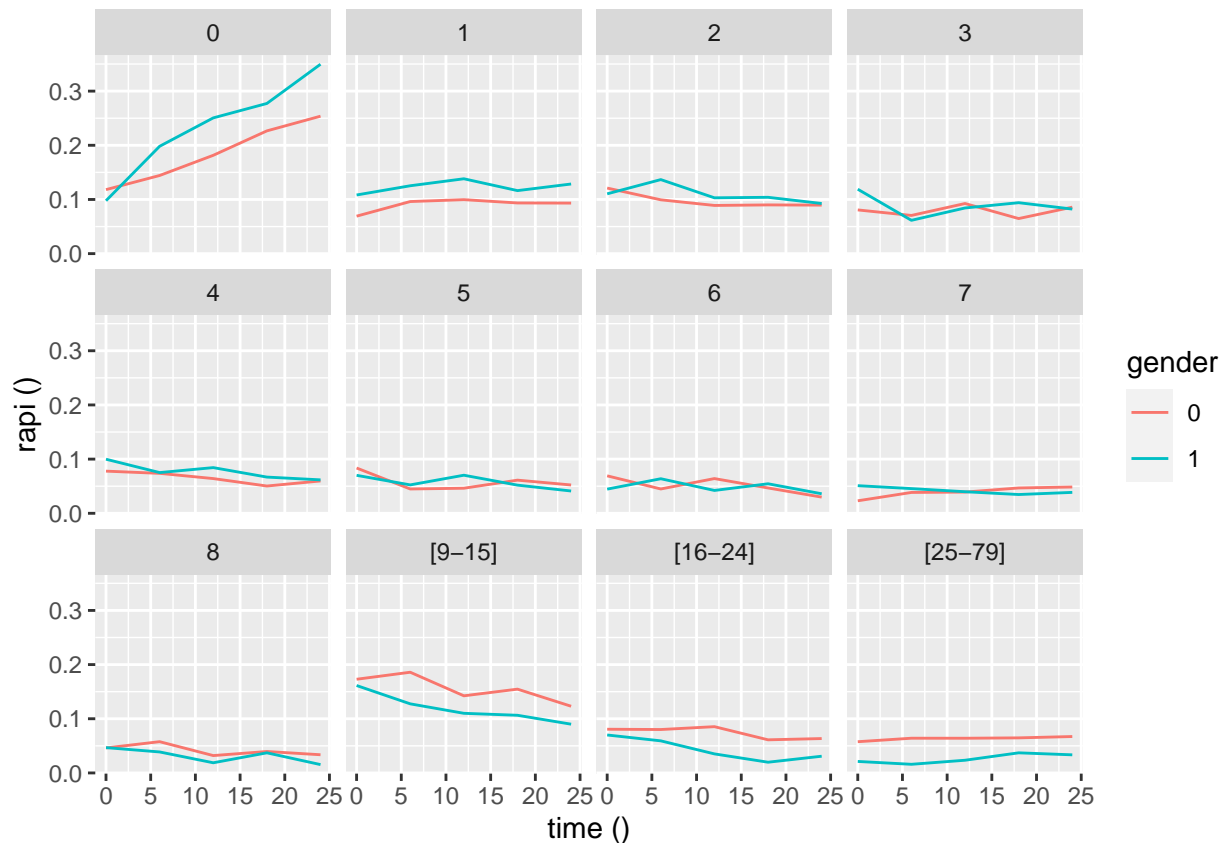
```

##
##
## The following SaemixData object was successfully created:
##
## Object of class SaemixData

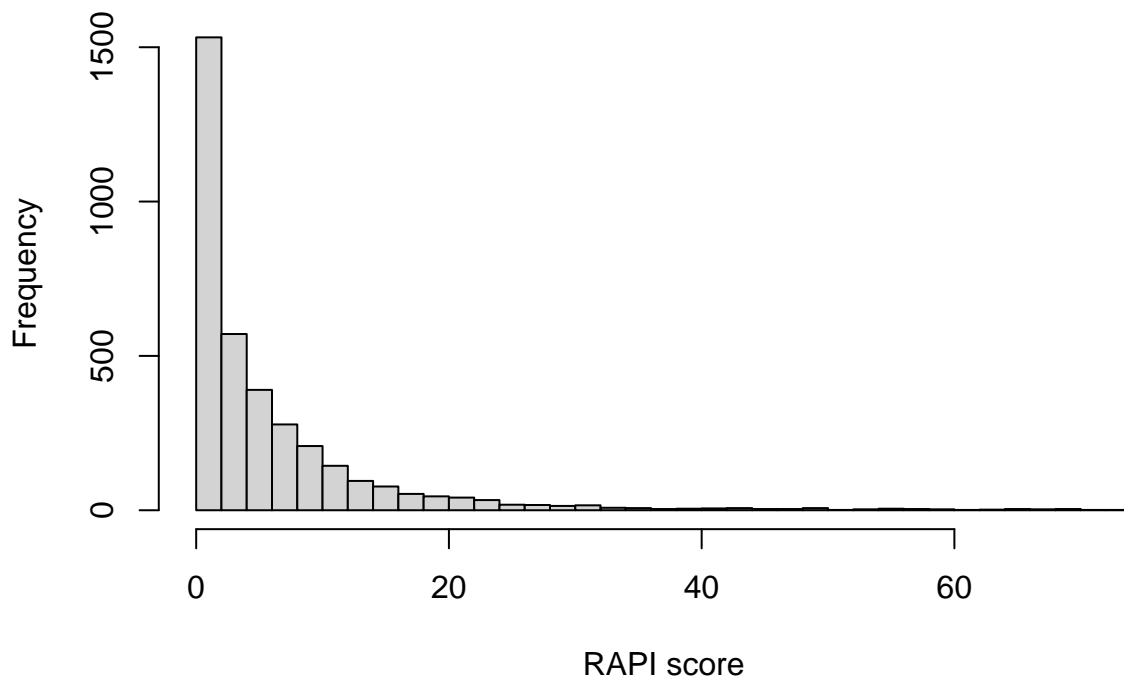
```

```
##      longitudinal data for use with the SAEM algorithm
## Dataset rapi.saemix
##      Structured data: rapi ~ time + rapi | id
##      X variable for graphs: time (months)
##      covariates: gender ()
##      reference class for covariate gender : Men

# Evolution of counts with time
xpl <- plotDiscreteData(saemix.data, outcome="count", which.cov="gender", breaks=c(0:9, 16, 25,80))
print(xpl)
```



```
# Histogram
hist(rapi.saemix$rapi, main="", xlab="RAPI score", breaks=30)
```



```
## Models
# Poisson with a time effect
count.poisson<-function(psi,id,xidep) {
  time<-xidep[,1]
  y<-xidep[,2]
  intercept<-psi[id,1]
  slope<-psi[id,2]
  lambda<- exp(intercept + slope*time)
  logp <- -lambda + y*log(lambda) - log(factorial(y))
  return(logp)
}
saemix.simulatePoisson<-function(psi, id, xidep) {
  time<-xidep[,1]
  y<-xidep[,2]
  intercept<-psi[id,1]
  slope<-psi[id,2]
  lambda<- exp(intercept + slope*time)
  y<-rpois(length(time), lambda=lambda)
  return(y)
}
# Fits
## Poisson
### Model without covariate
saemix.model.poi<-saemixModel(model=count.poisson,description="Count model Poisson",modeltype="likelihood",
  simulate.function=saemix.simulatePoisson,
  psi0=matrix(c(log(5),0.01),ncol=2,byrow=TRUE,dimnames=list(NULL, c("intercept","slope")),
  transform.par=c(0,0), omega.init=diag(c(0.5, 0.5)))

##
##
## The following SaemixModel object was successfully created:
##
```

```

## Nonlinear mixed-effects model
## Model function: Count model Poisson
## Model type: likelihood
## function(psi,id,xidep) {
##   time<-xidep[,1]
##   y<-xidep[,2]
##   intercept<-psi[id,1]
##   slope<-psi[id,2]
##   lambda<- exp(intercept + slope*time)
##   logp <- -lambda + y*log(lambda) - log(factorial(y))
##   return(logp)
## }
## Nb of parameters: 2
##   parameter names: intercept slope
##   distribution:
##   Parameter Distribution Estimated
## [1,] intercept normal      Estimated
## [2,] slope      normal      Estimated
## Variance-covariance matrix:
##           intercept slope
## intercept      1      0
## slope          0      1
## No covariate in the model.
## Initial values
##           intercept slope
## Pop.CondInit  1.609438  0.01

### Gender effect on intercept and slope
saemix.model.poi.cov2<-saemixModel(model=count.poisson,description="Count model Poisson",modeltype="lik
                                simulate.function=saemix.simulatePoisson,
                                psi0=matrix(c(log(5),0.01),ncol=2,byrow=TRUE,dimnames=list(NULL, c("
                                transform.par=c(0,0), omega.init=diag(c(0.5, 0.5)),
                                covariance.model =matrix(data=1, ncol=2, nrow=2),
                                covariate.model=matrix(c(1,1), ncol=2, byrow=TRUE))

##
##
## The following SaemixModel object was successfully created:
##
## Nonlinear mixed-effects model
## Model function: Count model Poisson
## Model type: likelihood
## function(psi,id,xidep) {
##   time<-xidep[,1]
##   y<-xidep[,2]
##   intercept<-psi[id,1]
##   slope<-psi[id,2]
##   lambda<- exp(intercept + slope*time)
##   logp <- -lambda + y*log(lambda) - log(factorial(y))
##   return(logp)
## }
## Nb of parameters: 2
##   parameter names: intercept slope
##   distribution:
##   Parameter Distribution Estimated

```

```

## [1,] intercept normal      Estimated
## [2,] slope      normal      Estimated
## Variance-covariance matrix:
##      intercept slope
## intercept      1      1
## slope           1      1
## Covariate model:
##      intercept slope
## [1,]           1      1
## Initial values
##      intercept slope
## Pop.CondInit  1.609438 0.01
## Cov.CondInit  0.000000 0.00

saemix.options<-list(seed=632545,save=FALSE,save.graphs=FALSE, displayProgress=FALSE, fim=FALSE)

### Fit with saemix
poisson.fit<-saemix(saemix.model.poi,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 rapi.saemix
## Structured data: rapi ~ time + rapi | id
## X variable for graphs: time (months)
## covariates: gender ()
## reference class for covariate gender : Men
## Dataset characteristics:
## number of subjects:      818
## number of observations: 3616
## average/min/max nb obs: 4.42 / 1 / 5
## First 10 lines of data:
## id time rapi rapi.1 gender mdv cens occ ytype
## 1  1   0   0      0   Men   0   0   1   1
## 2  1   6   0      0   Men   0   0   1   1
## 3  1  18   0      0   Men   0   0   1   1
## 4  2   0   3      3  Women   0   0   1   1
## 5  2   6   6      6  Women   0   0   1   1
## 6  2  12   5      5  Women   0   0   1   1
## 7  2  18   4      4  Women   0   0   1   1
## 8  2  24   5      5  Women   0   0   1   1
## 9  3   0   9      9   Men   0   0   1   1
## 10 3  12   1      1   Men   0   0   1   1
## -----
## ----          Model          ----
## -----
## Nonlinear mixed-effects model
## Model function: Count model Poisson
## Model type: likelihood
## function(psi,id,xidep) {
##   time<-xidep[,1]
##   y<-xidep[,2]

```

```

## intercept<-psi[id,1]
## slope<-psi[id,2]
## lambda<- exp(intercept + slope*time)
## logp <- -lambda + y*log(lambda) - log(factorial(y))
## return(logp)
## }
## <bytecode: 0x55c27c3e0288>
## Nb of parameters: 2
## parameter names: intercept slope
## distribution:
## Parameter Distribution Estimated
## [1,] intercept normal Estimated
## [2,] slope normal Estimated
## Variance-covariance matrix:
## intercept slope
## intercept 1 0
## slope 0 1
## No covariate in the model.
## Initial values
## intercept slope
## Pop.CondInit 1.609438 0.01
## -----
## ---- Key algorithm options ----
## -----
## Estimation of individual parameters (MAP)
## 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
## [1,] intercept 1.577
## [2,] slope -0.033
## -----
## ----- Variance of random effects -----
## -----
## Parameter Estimate
## intercept omega2.intercept 0.9039
## slope omega2.slope 0.0039
## -----
## ----- Correlation matrix of random effects -----
## -----
## omega2.intercept omega2.slope

```



```

## omega2.intercept 1          0
## omega2.slope     0          1
## -----
## ----- Statistical criteria -----
## -----
##
## Likelihood computed by importance sampling
##      -2LL= 21486.75
##      AIC = 21496.75
##      BIC = 21520.29
## -----

poisson.fit.cov2<-saemix(saemix.model.poi.cov2,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 rapi.saemix
##      Structured data: rapi ~ time + rapi | id
##      X variable for graphs: time (months)
##      covariates: gender ()
##      reference class for covariate gender : Men
## Dataset characteristics:
##      number of subjects:      818
##      number of observations: 3616
##      average/min/max nb obs: 4.42 / 1 / 5
## First 10 lines of data:
##      id time rapi rapi.1 gender mdv cens occ ytype
## 1  1  0  0  0  Men  0  0  1  1
## 2  1  6  0  0  Men  0  0  1  1
## 3  1 18  0  0  Men  0  0  1  1
## 4  2  0  3  3  Women 0  0  1  1
## 5  2  6  6  6  Women 0  0  1  1
## 6  2 12  5  5  Women 0  0  1  1
## 7  2 18  4  4  Women 0  0  1  1
## 8  2 24  5  5  Women 0  0  1  1
## 9  3  0  9  9  Men  0  0  1  1
## 10 3 12  1  1  Men  0  0  1  1
## -----
## ---- Model ----
## -----
## Nonlinear mixed-effects model
##      Model function: Count model Poisson
##      Model type: likelihood
## function(psi,id,xidep) {
##      time<-xidep[,1]
##      y<-xidep[,2]
##      intercept<-psi[id,1]
##      slope<-psi[id,2]
##      lambda<- exp(intercept + slope*time)
##      logp <- -lambda + y*log(lambda) - log(factorial(y))
##      return(logp)

```

```

## }
## <bytecode: 0x55c27c3e0288>
##   Nb of parameters: 2
##     parameter names:  intercept slope
##     distribution:
##       Parameter Distribution Estimated
## [1,] intercept normal      Estimated
## [2,] slope      normal      Estimated
##   Variance-covariance matrix:
##     intercept slope
## intercept      1      1
## slope          1      1
##   Covariate model:
##     [,1] [,2]
## gender   1      1
##   Initial values
##     intercept slope
## Pop.CondInit  1.609438  0.01
## Cov.CondInit  0.000000  0.00
## -----
## ----   Key algorithm options   ----
## -----
##   Estimation of individual parameters (MAP)
##   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
## [1,] intercept      1.683
## [2,] beta_gender(intercept) -0.196
## [3,] slope          -0.022
## [4,] beta_gender(slope)   -0.017
## -----
## ----- Variance of random effects -----
## -----
##   Parameter      Estimate
## intercept omega2.intercept 0.9179
## slope      omega2.slope    0.0039
## -----
## ----- Correlation matrix of random effects -----
## -----
##               omega2.intercept omega2.slope

```

```
## omega2.intercept  1.00          -0.14
## omega2.slope     -0.14          1.00
## -----
## ----- Statistical criteria -----
## -----
##
## Likelihood computed by importance sampling
##      -2LL= 21454.94
##      AIC = 21470.94
##      BIC = 21508.59
## -----
```

```
exp(poisson.fit@results@fixed.effects)
```

```
## [1] 4.8394604 0.9673886
```

```
exp(poisson.fit.cov2@results@fixed.effects)
```

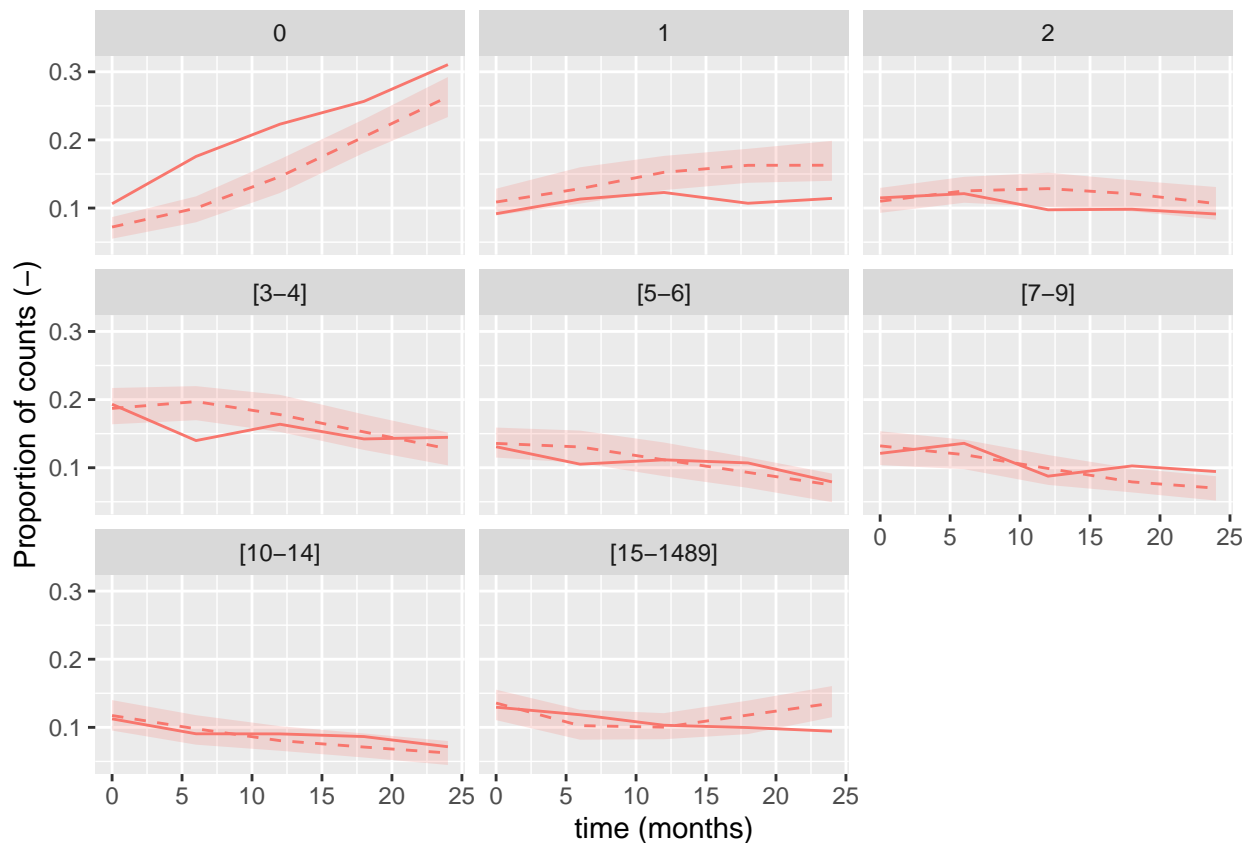
```
## [1] 5.3842360 0.8217414 0.9780800 0.9833256
```

```
### Simulations
```

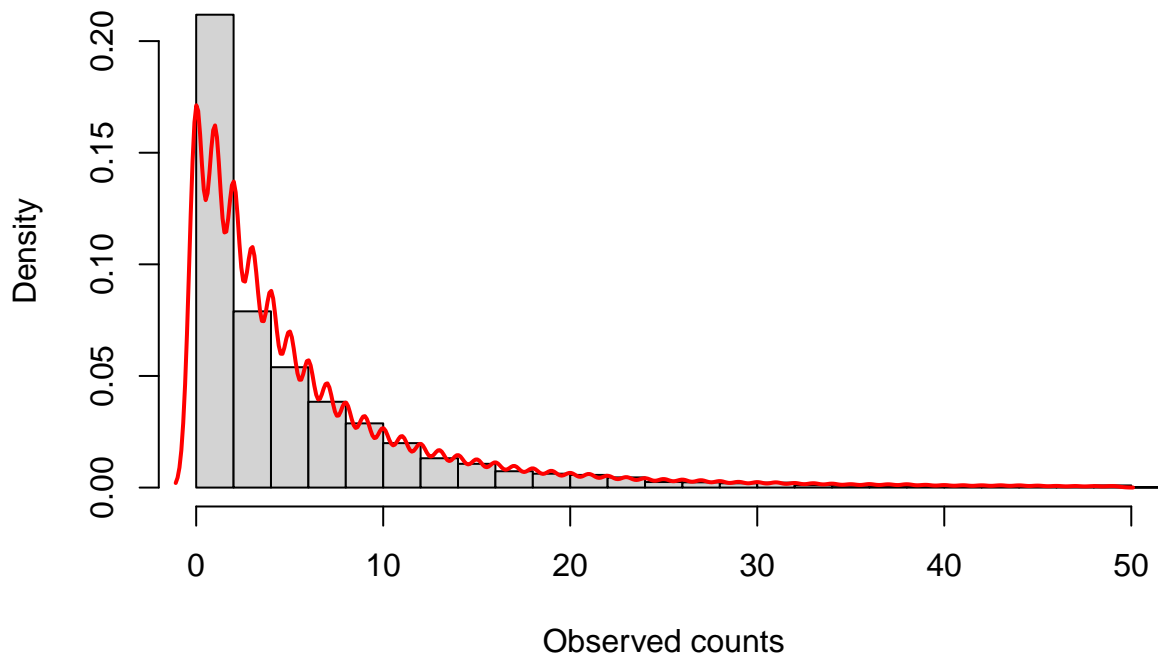
```
nsim<-100
```

```
yfit1<-simulateDiscreteSaemix(poisson.fit.cov2, nsim=nsim)
```

```
discreteVPC(yfit1, outcome="count")
```



```
hist(yfit1@data@data$rapi, xlim=c(0,50), freq=F, breaks=30, xlab="Observed counts", main="")
lines(density(yfit1@sim.data@datasim$ysim[yfit1@sim.data@datasim$ysim<50]), lwd = 2, col = 'red')
```



```
cat("Observed proportion of 0's", length(yfit1@data@data$rap1[yfit1@data@data$rap1==0])/yfit1@data@ntot)
```

```
## Observed proportion of 0's 0.2090708
```

```
cat("      Poisson model, p=", length(yfit1@sim.data@datasim$ysim[yfit1@sim.data@datasim$ysim==0])/length(yfit1@sim.data@datasim$ysim))
```

```
##      Poisson model, p= 0.1518501
```

```
# Checking proportion of zeroes
```

```
yfit<-yfit1
```

```
simdat <-yfit@sim.data@datasim
```

```
simdat$time<-rep(yfit@data@data$time,nsim)
```

```
simdat$gender<-rep(yfit@data@data$gender,nsim)
```

```
ytab<-NULL
```

```
for(irep in 1:nsim) {
```

```
  xtab<-simdat[simdat$irep==irep,]
```

```
  suppressMessages(
```

```
    xtab1 <- xtab %>%
```

```
      group_by(time, gender) %>%
```

```
      summarise(nev = sum(ysim==0), n=n()) %>%
```

```
      mutate(freq = nev/n)
```

```
  )
```

```
  ytab<-rbind(ytab,xtab1[,c("time","gender","freq")])
```

```
}
```

```
gtab <- ytab %>%
```

```
  group_by(time, gender) %>%
```

```
  summarise(lower=quantile(freq, c(0.05)), median=quantile(freq, c(0.5)), upper=quantile(freq, c(0.95)))
```

```
  mutate(gender=ifelse(gender==0,"Men","Women"))
```

```
## `summarise()` has grouped output by 'time'. You can override using the
```

```
## `.groups` argument.
```

```
gtab$freq<-1
```

```
gtab1<-cbind(gtab, model="Poisson")
```

```

rapipl <- rapi.saemix %>%
  group_by(time, gender) %>%
  summarise(nev = sum(rapi==0), n=n()) %>%
  mutate(freq = nev/n, sd=sqrt((1-nev/n)/nev)) %>%
  mutate(lower=freq-1.96*sd, upper=freq+1.96*sd)

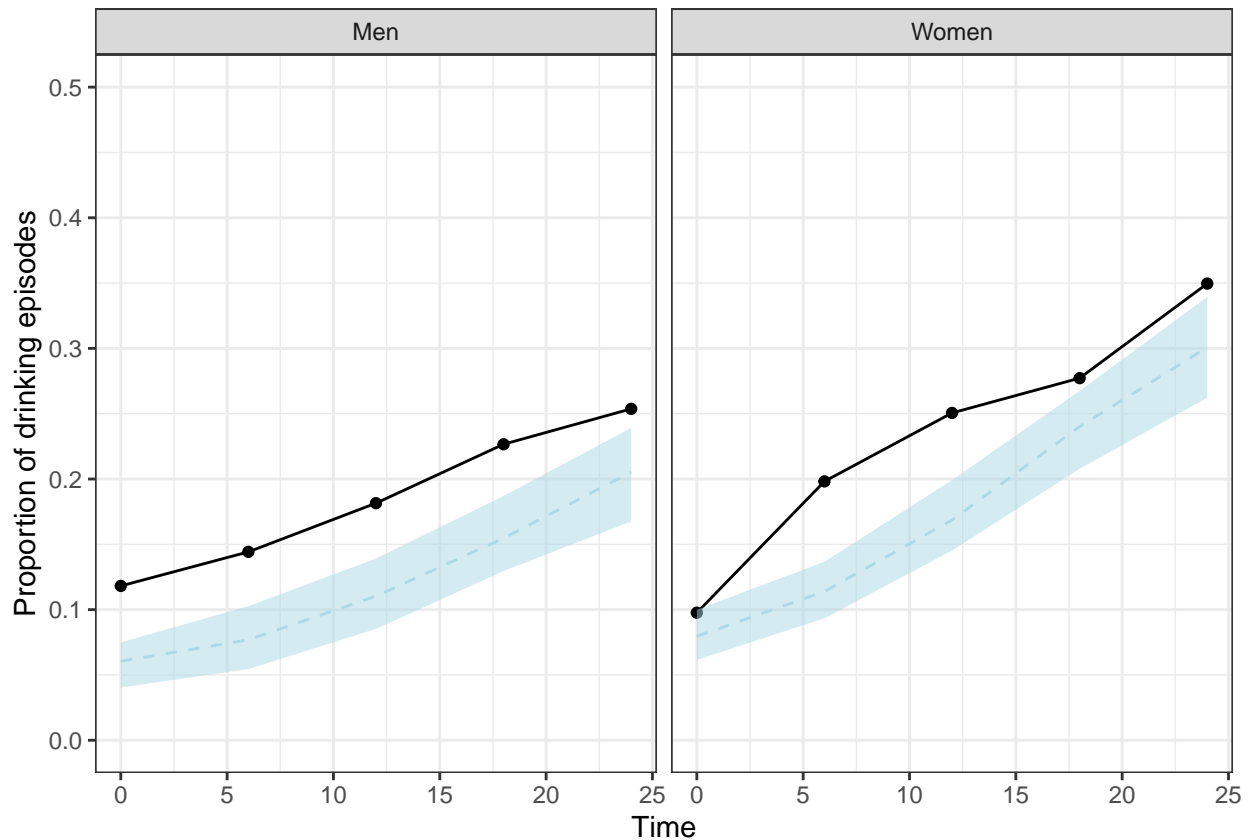
## `summarise()` has grouped output by 'time'. You can override using the
## `.groups` argument.

rapipl$lower[rapipl$lower<0] <-0 # we should use a better approximation for CI

plot2 <- ggplot(rapipl, aes(x=time, y=freq, group=gender)) + geom_line() +
  geom_point() +
  geom_line(data=gtab, aes(x=time, y=median, group=gender), linetype=2, colour='lightblue') +
  geom_ribbon(data=gtab, aes(ymin=lower, ymax=upper, group=gender), alpha=0.5, fill='lightblue') +
  ylim(c(0,0.5)) + theme_bw() + theme(legend.position = "none") + facet_wrap(~gender) +
  xlab("Time") + ylab("Proportion of drinking episodes")

print(plot2)

```



```

## ZIP Poisson model with time effect
count.poissonzip<-function(psi,id,xidep) {
  time<-xidep[,1]
  y<-xidep[,2]
  intercept<-psi[id,1]
  slope<-psi[id,2]

```

```

p0<-psi[id,3] # Probability of zero's
lambda<- exp(intercept + slope*time)
logp <- log(1-p0) -lambda + y*log(lambda) - log(factorial(y)) # Poisson
logp0 <- log(p0+(1-p0)*exp(-lambda)) # Zeroes
logp[y==0]<-logp0[y==0]
return(logp)
}

## Generalized Poisson model with time effect
count.genpoisson<-function(psi,id,xidep) {
  time<-xidep[,1]
  y<-xidep[,2]
  intercept<-psi[id,1]
  slope<-psi[id,2]
  lambda<- exp(intercept + slope*time)
  delta<-psi[id,3]
  logp <- log(lambda) + (y-1)*log(lambda+y*delta) - lambda - y*delta - log(factorial(y))
  return(logp)
}

## Negative binomial model with time effect
count.NB<-function(psi,id,xidep) {
  time<-xidep[,1]
  y<-xidep[,2]
  intercept<-psi[id,1]
  slope<-psi[id,2]
  k<-psi[id,3]
  lambda<- exp(intercept + slope*time)
  logp <- log(factorial(y+k-1)) - log(factorial(y)) - log(factorial(k-1)) + y*log(lambda) - y*log(lambda)
  return(logp)
}

saemix.simulatePoissonZIP<-function(psi, id, xidep) {
  time<-xidep[,1]
  y<-xidep[,2]
  intercept<-psi[id,1]
  slope<-psi[id,2]
  p0<-psi[id,3] # Probability of zero's
  lambda<- exp(intercept + slope*time)
  prob0<-rbinom(length(time), size=1, prob=p0)
  y<-rpois(length(time), lambda=lambda)
  y[prob0==1]<-0
  return(y)
}

## ZIP
### base model
saemix.model.zip<-saemixModel(model=count.poissonzip,description="count model ZIP",modeltype="likelihood",
  simulate.function=saemix.simulatePoissonZIP,
  psi0=matrix(c(1.5, 0.01, 0.2),ncol=3,byrow=TRUE,dimnames=list(NULL, c("intercept", "slope", "delta")),
  transform.par=c(0,0,3), covariance.model=diag(c(1,1,0)), omega.init=diag(

```

Overdispersion

```
##
##
```

```

## The following SaemixModel object was successfully created:
##
## Nonlinear mixed-effects model
##   Model function:  count model ZIP
##   Model type:  likelihood
## function(psi,id,xidep) {
##   time<-xidep[,1]
##   y<-xidep[,2]
##   intercept<-psi[id,1]
##   slope<-psi[id,2]
##   p0<-psi[id,3] # Probability of zero's
##   lambda<- exp(intercept + slope*time)
##   logp <- log(1-p0) -lambda + y*log(lambda) - log(factorial(y)) # Poisson
##   logp0 <- log(p0+(1-p0)*exp(-lambda)) # Zeroes
##   logp[y==0]<-logp0[y==0]
##   return(logp)
## }
##   Nb of parameters: 3
##       parameter names:  intercept slope p0
##       distribution:
##       Parameter Distribution Estimated
## [1,] intercept normal      Estimated
## [2,] slope      normal      Estimated
## [3,] p0          logit      Estimated
##   Variance-covariance matrix:
##           intercept slope p0
## intercept      1      0  0
## slope          0      1  0
## p0             0      0  0
##   No covariate in the model.
##   Initial values
##           intercept slope  p0
## Pop.CondInit      1.5  0.01 0.2

#### ZIP Poisson with gender on both intercept
saemix.model.zip.cov1<-saemixModel(model=count.poissonzip,description="count model ZIP",modeltype="like",
                                simulate.function=saemix.simulatePoissonZIP,
                                psi0=matrix(c(1.5, 0.01, 0.2),ncol=3,byrow=TRUE,dimnames=list(NULL, c("intercept", "slope", "p0")),
                                transform.par=c(0,0,3), covariance.model=diag(c(1,1,0)), omega.init=c(0,0,0),
                                covariate.model = matrix(c(1,0,0),ncol=3, byrow=TRUE))

##
##
## The following SaemixModel object was successfully created:
##
## Nonlinear mixed-effects model
##   Model function:  count model ZIP
##   Model type:  likelihood
## function(psi,id,xidep) {
##   time<-xidep[,1]
##   y<-xidep[,2]
##   intercept<-psi[id,1]
##   slope<-psi[id,2]
##   p0<-psi[id,3] # Probability of zero's
##   lambda<- exp(intercept + slope*time)

```

```

## logp <- log(1-p0) -lambda + y*log(lambda) - log(factorial(y)) # Poisson
## logp0 <- log(p0+(1-p0)*exp(-lambda)) # Zeroes
## logp[y==0]<-logp0[y==0]
## return(logp)
## }
## Nb of parameters: 3
## parameter names: intercept slope p0
## distribution:
## Parameter Distribution Estimated
## [1,] intercept normal Estimated
## [2,] slope normal Estimated
## [3,] p0 logit Estimated
## Variance-covariance matrix:
## intercept slope p0
## intercept 1 0 0
## slope 0 1 0
## p0 0 0 0
## Covariate model:
## intercept slope p0
## [1,] 1 0 0
## Initial values
## intercept slope p0
## Pop.CondInit 1.5 0.01 0.2
## Cov.CondInit 0.0 0.00 0.0

#### ZIP Poisson with gender on both intercept and slope
saemix.model.zip.cov2<-saemixModel(model=count.poissonzip,description="count model ZIP",modeltype="likelihood",
simulate.function=saemix.simulatePoissonZIP,
psi0=matrix(c(1.5, 0.01, 0.2),ncol=3,byrow=TRUE,dimnames=list(NULL, c("intercept", "slope", "p0"))),
transform.par=c(0,0,3), covariance.model=diag(c(1,1,0)), omega.init=matrix(c(1,1,0),ncol=3, byrow=TRUE))

##
##
## The following SaemixModel object was successfully created:
##
## Nonlinear mixed-effects model
## Model function: count model ZIP
## Model type: likelihood
## function(psi,id,xidep) {
## time<-xidep[,1]
## y<-xidep[,2]
## intercept<-psi[id,1]
## slope<-psi[id,2]
## p0<-psi[id,3] # Probability of zero's
## lambda<- exp(intercept + slope*time)
## logp <- log(1-p0) -lambda + y*log(lambda) - log(factorial(y)) # Poisson
## logp0 <- log(p0+(1-p0)*exp(-lambda)) # Zeroes
## logp[y==0]<-logp0[y==0]
## return(logp)
## }
## Nb of parameters: 3
## parameter names: intercept slope p0
## distribution:
## Parameter Distribution Estimated

```



```
## [1,] intercept normal      Estimated
## [2,] slope      normal      Estimated
## [3,] p0         logit        Estimated
## Variance-covariance matrix:
##      intercept slope p0
## intercept      1    0  0
## slope          0    1  0
## p0             0    0  0
## Covariate model:
##      intercept slope p0
## [1,]          1    1  0
## Initial values
##      intercept slope p0
## Pop.CondInit      1.5 0.01 0.2
## Cov.CondInit      0.0 0.00 0.0

zippoisson.fit<-saemix(saemix.model.zip,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 rapi.saemix
##      Structured data: rapi ~ time + rapi | id
##      X variable for graphs: time (months)
##      covariates: gender ()
##      reference class for covariate gender : Men
## Dataset characteristics:
##      number of subjects:      818
##      number of observations: 3616
##      average/min/max nb obs: 4.42 / 1 / 5
## First 10 lines of data:
##      id time rapi rapi.1 gender mdv cens occ ytype
## 1  1    0    0    0    Men    0    0    1    1
## 2  1    6    0    0    Men    0    0    1    1
## 3  1   18    0    0    Men    0    0    1    1
## 4  2    0    3    3  Women    0    0    1    1
## 5  2    6    6    6  Women    0    0    1    1
## 6  2   12    5    5  Women    0    0    1    1
## 7  2   18    4    4  Women    0    0    1    1
## 8  2   24    5    5  Women    0    0    1    1
## 9  3    0    9    9    Men    0    0    1    1
## 10 3   12    1    1    Men    0    0    1    1
## -----
## ----          Model          ----
## -----
## Nonlinear mixed-effects model
##      Model function: count model ZIP
##      Model type: likelihood
## function(psi,id,xidep) {
##      time<-xidep[,1]
##      y<-xidep[,2]
##      intercept<-psi[id,1]
```

```

## slope<-psi[id,2]
## p0<-psi[id,3] # Probability of zero's
## lambda<- exp(intercept + slope*time)
## logp <- log(1-p0) -lambda + y*log(lambda) - log(factorial(y)) # Poisson
## logp0 <- log(p0+(1-p0)*exp(-lambda)) # Zeroes
## logp[y==0]<-logp0[y==0]
## return(logp)
## }
## <bytecode: 0x55c276db8d78>
## Nb of parameters: 3
## parameter names: intercept slope p0
## distribution:
## Parameter Distribution Estimated
## [1,] intercept normal Estimated
## [2,] slope normal Estimated
## [3,] p0 logit Estimated
## Variance-covariance matrix:
## intercept slope p0
## intercept 1 0 0
## slope 0 1 0
## p0 0 0 0
## No covariate in the model.
## Initial values
## intercept slope p0
## Pop.CondInit 1.5 0.01 0.2
## -----
## ---- Key algorithm options ----
## -----
## Estimation of individual parameters (MAP)
## 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
## [1,] intercept 1.657
## [2,] slope -0.029
## [3,] p0 0.076
## -----
## ----- Variance of random effects -----
## -----
## Parameter Estimate
## intercept omega2.intercept 0.7977

```

```

## slope      omega2.slope      0.0032
## -----
## ----- Correlation matrix of random effects -----
## -----
##              omega2.intercept omega2.slope
## omega2.intercept 1              0
## omega2.slope     0              1
## -----
## ----- Statistical criteria -----
## -----
##
## Likelihood computed by importance sampling
##      -2LL= 20479.88
##      AIC = 20491.88
##      BIC = 20520.12
## -----
zippoisson.fit.cov1<-saemix(saemix.model.zip.cov1,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 rapi.saemix
##      Structured data: rapi ~ time + rapi | id
##      X variable for graphs: time (months)
##      covariates: gender ()
##      reference class for covariate gender : Men
## Dataset characteristics:
##      number of subjects:      818
##      number of observations: 3616
##      average/min/max nb obs: 4.42 / 1 / 5
## First 10 lines of data:
##      id time rapi rapi.1 gender mdv cens occ ytype
## 1  1  0  0  0  Men  0  0  1  1
## 2  1  6  0  0  Men  0  0  1  1
## 3  1 18  0  0  Men  0  0  1  1
## 4  2  0  3  3  Women 0  0  1  1
## 5  2  6  6  6  Women 0  0  1  1
## 6  2 12  5  5  Women 0  0  1  1
## 7  2 18  4  4  Women 0  0  1  1
## 8  2 24  5  5  Women 0  0  1  1
## 9  3  0  9  9  Men  0  0  1  1
## 10 3 12  1  1  Men  0  0  1  1
## -----
## ----- Model -----
## -----
## Nonlinear mixed-effects model
##      Model function: count model ZIP
##      Model type: likelihood
## function(psi,id,xidep) {
##      time<-xidep[,1]
##      y<-xidep[,2]

```

```

## intercept<-psi[id,1]
## slope<-psi[id,2]
## p0<-psi[id,3] # Probability of zero's
## lambda<- exp(intercept + slope*time)
## logp <- log(1-p0) -lambda + y*log(lambda) - log(factorial(y)) # Poisson
## logp0 <- log(p0+(1-p0)*exp(-lambda)) # Zeroes
## logp[y==0]<-logp0[y==0]
## return(logp)
## }
## <bytecode: 0x55c276db8d78>
## Nb of parameters: 3
## parameter names: intercept slope p0
## distribution:
## Parameter Distribution Estimated
## [1,] intercept normal Estimated
## [2,] slope normal Estimated
## [3,] p0 logit Estimated
## Variance-covariance matrix:
## intercept slope p0
## intercept 1 0 0
## slope 0 1 0
## p0 0 0 0
## Covariate model:
## [,1] [,2] [,3]
## gender 1 0 0
## Initial values
## intercept slope p0
## Pop.CondInit 1.5 0.01 0.2
## Cov.CondInit 0.0 0.00 0.0
## -----
## ---- Key algorithm options ----
## -----
## Estimation of individual parameters (MAP)
## 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
## [1,] intercept 1.786
## [2,] beta_gender(intercept) -0.226
## [3,] slope -0.029
## [4,] p0 0.076

```

```

## -----
## ----- Variance of random effects -----
## -----
##           Parameter           Estimate
## intercept omega2.intercept 0.7849
## slope      omega2.slope    0.0033
## -----
## ----- Correlation matrix of random effects -----
## -----
##           omega2.intercept omega2.slope
## omega2.intercept 1           0
## omega2.slope     0           1
## -----
## ----- Statistical criteria -----
## -----
## Likelihood computed by importance sampling
##      -2LL= 20469.41
##      AIC = 20483.41
##      BIC = 20516.35
## -----

zippoisson.fit.cov2<-saemix(saemix.model.zip.cov2,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 rapi.saemix
##      Structured data: rapi ~ time + rapi | id
##      X variable for graphs: time (months)
##      covariates: gender ()
##      reference class for covariate gender : Men
## Dataset characteristics:
##      number of subjects:      818
##      number of observations: 3616
##      average/min/max nb obs: 4.42 / 1 / 5
## First 10 lines of data:
##      id time rapi rapi.1 gender mdv cens occ ytype
## 1  1  0  0  0  Men  0  0  1  1
## 2  1  6  0  0  Men  0  0  1  1
## 3  1 18  0  0  Men  0  0  1  1
## 4  2  0  3  3 Women  0  0  1  1
## 5  2  6  6  6 Women  0  0  1  1
## 6  2 12  5  5 Women  0  0  1  1
## 7  2 18  4  4 Women  0  0  1  1
## 8  2 24  5  5 Women  0  0  1  1
## 9  3  0  9  9  Men  0  0  1  1
## 10 3 12  1  1  Men  0  0  1  1
## -----
## ----- Model -----
## -----
## Nonlinear mixed-effects model

```

```

## Model function: count model ZIP
## Model type: likelihood
## function(psi,id,xidep) {
##   time<-xidep[,1]
##   y<-xidep[,2]
##   intercept<-psi[id,1]
##   slope<-psi[id,2]
##   p0<-psi[id,3] # Probability of zero's
##   lambda<- exp(intercept + slope*time)
##   logp <- log(1-p0) -lambda + y*log(lambda) - log(factorial(y)) # Poisson
##   logp0 <- log(p0+(1-p0)*exp(-lambda)) # Zeroes
##   logp[y==0]<-logp0[y==0]
##   return(logp)
## }
## <bytecode: 0x55c276db8d78>
## Nb of parameters: 3
##   parameter names: intercept slope p0
##   distribution:
##   Parameter Distribution Estimated
## [1,] intercept normal Estimated
## [2,] slope normal Estimated
## [3,] p0 logit Estimated
## Variance-covariance matrix:
##   intercept slope p0
## intercept 1 0 0
## slope 0 1 0
## p0 0 0 0
## Covariate model:
##   [,1] [,2] [,3]
## gender 1 1 0
## Initial values
##   intercept slope p0
## Pop.CondInit 1.5 0.01 0.2
## Cov.CondInit 0.0 0.00 0.0
## -----
## ---- Key algorithm options ----
## -----
## Estimation of individual parameters (MAP)
## 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
## [1,] intercept      1.773
## [2,] beta_gender(intercept) -0.197
## [3,] slope          -0.020
## [4,] beta_gender(slope)   -0.016
## [5,] p0              0.075
## -----
## ----- Variance of random effects -----
## -----
##      Parameter      Estimate
## intercept omega2.intercept 0.7826
## slope      omega2.slope    0.0033
## -----
## ----- Correlation matrix of random effects -----
## -----
##      omega2.intercept omega2.slope
## omega2.intercept 1      0
## omega2.slope     0      1
## -----
## ----- Statistical criteria -----
## -----
##
## Likelihood computed by importance sampling
##      -2LL= 20459.27
##      AIC = 20475.27
##      BIC = 20512.93
## -----
```

```
exp(zippoisson.fit@results@fixed.effects)
```

```
## [1] 5.2450012 0.9714983 1.0793068
```

```
exp(zippoisson.fit.cov1@results@fixed.effects)
```

```
## [1] 5.9656256 0.7975888 0.9714754 1.0793259
```

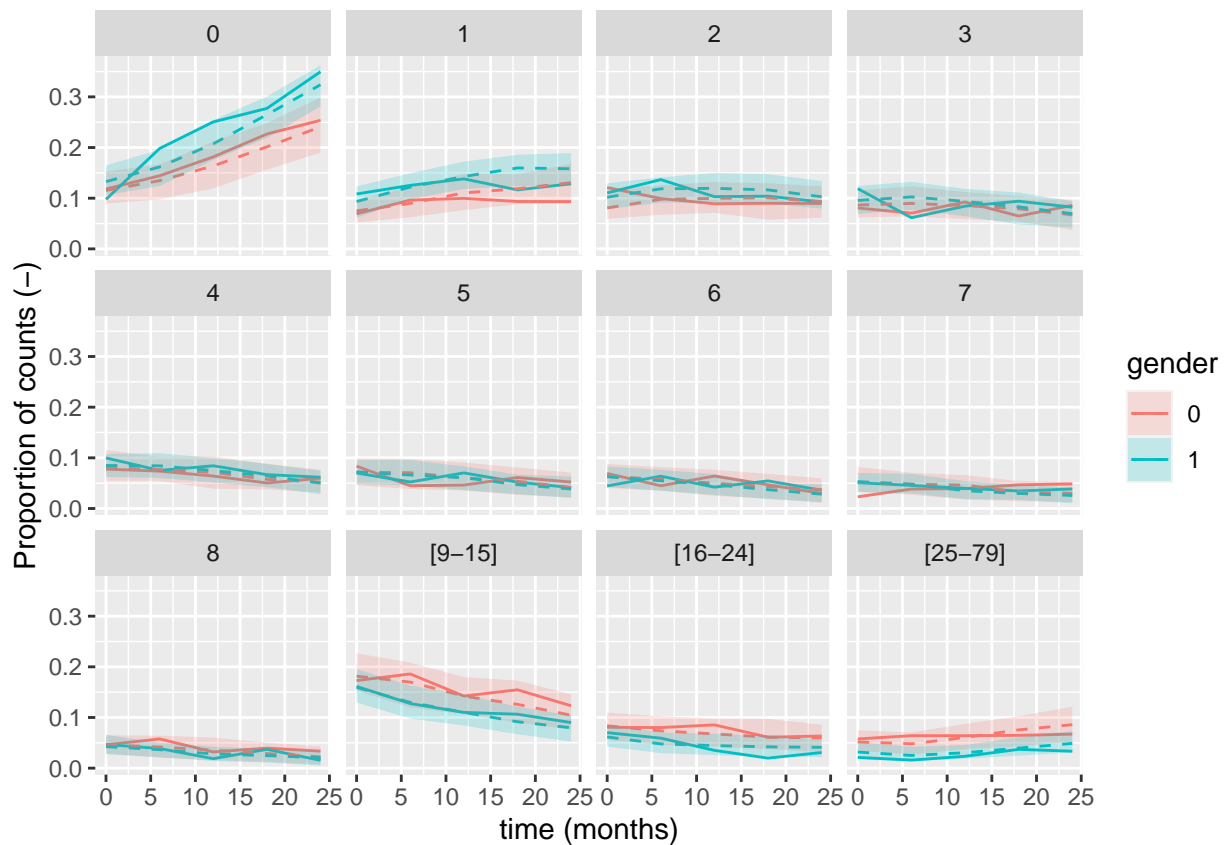
```
exp(zippoisson.fit.cov2@results@fixed.effects)
```

```
## [1] 5.8872267 0.8213610 0.9797720 0.9844017 1.0783237
```

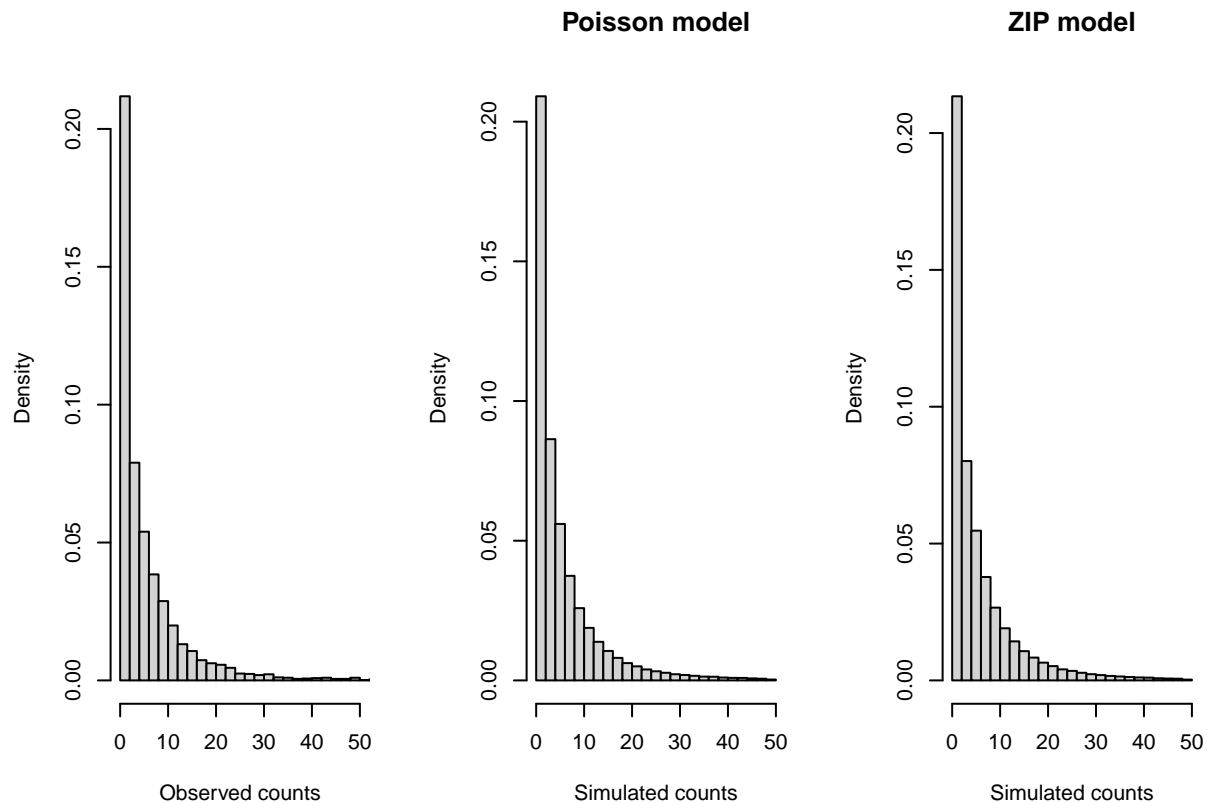
```
### Simulations
```

```
yfit2<-simulateDiscreteSaemix(zippoisson.fit.cov2, 100)
```

```
discreteVPC(yfit2, outcome="count", which.cov="gender", breaks=c(0:9, 16, 25,80))
```



```
cat("Observed proportion of 0's", length(yfit1@data@data$rap1[yfit1@data@data$rap1==0])/yfit1@data@ntot)
## Observed proportion of 0's 0.2090708
cat("      Poisson model, p=", length(yfit1@sim.data@datasim$ysim[yfit1@sim.data@datasim$ysim==0])/length(yfit1@sim.data@ntot))
##      Poisson model, p= 0.1518501
cat("      ZI-Poisson model, p=", length(yfit2@sim.data@datasim$ysim[yfit2@sim.data@datasim$ysim==0])/length(yfit2@sim.data@ntot))
##      ZI-Poisson model, p= 0.1957329
par(mfrow=c(1,3))
hist(yfit1@data@data$rap1, xlim=c(0,50), freq=F, breaks=30, xlab="Observed counts", main="")
hist(yfit1@sim.data@datasim$ysim[yfit1@sim.data@datasim$ysim<50], xlim=c(0,50), freq=F, breaks=20, xlab="Poisson model")
hist(yfit2@sim.data@datasim$ysim[yfit2@sim.data@datasim$ysim<50], xlim=c(0,50), freq=F, breaks=20, xlab="ZI-Poisson model")
```

```
# Checking proportion of zeroes
yfit<-yfit2
simdat <-yfit@sim.data@datasim
simdat$time<-rep(yfit@data@data$time,nsim)
simdat$gender<-rep(yfit@data@data$gender,nsim)

ytab<-NULL
for(irep in 1:nsim) {
  xtab<-simdat[simdat$irep==irep,]
  suppressMessages(
    xtab1 <- xtab %>%
      group_by(time, gender) %>%
      summarise(nev = sum(ysim==0), n=n()) %>%
      mutate(freq = nev/n)
  )
  ytab<-rbind(ytab,xtab1[,c("time","gender","freq")])
}
gtab <- ytab %>%
  group_by(time, gender) %>%
  summarise(lower=quantile(freq, c(0.05)), median=quantile(freq, c(0.5)), upper=quantile(freq, c(0.95)))
  mutate(gender=ifelse(gender==0,"Men","Women"))

## `summarise()` has grouped output by 'time'. You can override using the
## `groups` argument.

gtab$freq<-1
gtab2<-cbind(gtab, model="ZIP")
gtab<-rbind(gtab1, gtab2)
```

```

rapi1 <- rapi.saemix %>%
  group_by(time, gender) %>%
  summarise(nev = sum(rapi==0), n=n()) %>%
  mutate(freq = nev/n, sd=sqrt((1-nev/n)/nev)) %>%
  mutate(lower=freq-1.96*sd, upper=freq+1.96*sd)

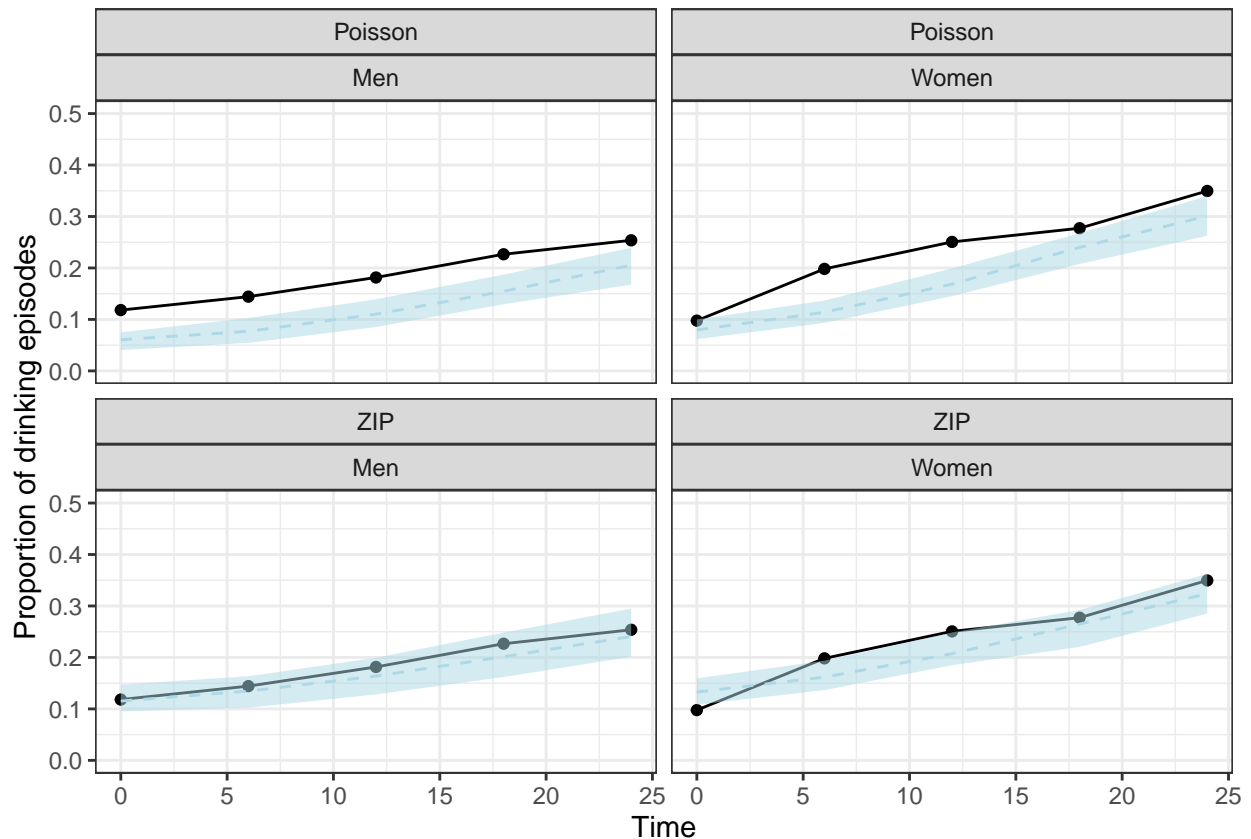
## `summarise()` has grouped output by 'time'. You can override using the
## `.groups` argument.

rapi1$lower[rapi1$lower<0] <-0 # we should use a better approximation for CI

plot2 <- ggplot(rapi1, aes(x=time, y=freq, group=gender)) + geom_line() +
  geom_point() +
  geom_line(data=gtab, aes(x=time, y=median, group=gender), linetype=2, colour='lightblue') +
  geom_ribbon(data=gtab, aes(ymin=lower, ymax=upper, group=gender), alpha=0.5, fill='lightblue') +
  ylim(c(0,0.5)) + theme_bw() + theme(legend.position = "none") + facet_wrap(model~gender) +
  xlab("Time") + ylab("Proportion of drinking episodes")

print(plot2)

```



```

## Hurdle - 2 models
saemix.data1<-saemixData(name.data=rapi.saemix[rapi.saemix$rapi>0,], name.group=c("id"),
  name.predictors=c("time","rapi"),name.response=c("rapi"),
  name.covariates=c("gender"),
  units=list(x="week",y="",covariates=c("")))

```

Hurdle model

```
##
##
## The following SaemixData object was successfully created:
##
## Object of class SaemixData
##   longitudinal data for use with the SAEM algorithm
## Dataset rapi.saemix[rapi.saemix$rapi > 0, ]
##   Structured data: rapi ~ time + rapi | id
##   X variable for graphs: time (week)
##   covariates: gender ()
##   reference class for covariate gender : Men
rapi.saemix$y0<-as.integer(rapi.saemix$rapi==0)
saemix.data0<-saemixData(name.data=rapi.saemix, name.group=c("id"),
                        name.predictors=c("time","y0"),name.response=c("y0"),
                        name.covariates=c("gender"),
                        units=list(x="week",y="",covariates=c("")))

##
##
## The following SaemixData object was successfully created:
##
## Object of class SaemixData
##   longitudinal data for use with the SAEM algorithm
## Dataset rapi.saemix
##   Structured data: y0 ~ time + y0 | id
##   X variable for graphs: time (week)
##   covariates: gender ()
##   reference class for covariate gender : Men
# Fit Binomial model to saemix.data0
binary.model<-function(psi,id,xidep) {
  tim<-xidep[,1]
  y<-xidep[,2]
  inter<-psi[id,1]
  slope<-psi[id,2]
  logit<-inter+slope*tim
  pevent<-exp(logit)/(1+exp(logit))
  pobs = (y==0)*(1-pevent)+(y==1)*pevent
  logpdf <- log(pobs)
  return(logpdf)
}
simulBinary<-function(psi,id,xidep) {
  tim<-xidep[,1]
  y<-xidep[,2]
  inter<-psi[id,1]
  slope<-psi[id,2]
  logit<-inter+slope*tim
  pevent<-exp(logit)/(1+exp(logit))
  ysim<-rbinom(length(tim),size=1, prob=pevent)
  return(ysim)
}
saemix.hurdle0<-saemixModel(model=binary.model,description="Binary model",
                           modeltype="likelihood",simulate.function=simulBinary,
```

```

psi0=matrix(c(-1.5,-.1,0,0),ncol=2,byrow=TRUE,dimnames=list(NULL,c("theta1"
transform.par=c(0,0), covariate.model=c(1,1),
covariance.model=matrix(c(1,0,0,1),ncol=2), omega.init=diag(c(1,0.3)))

##
##
## The following SaemixModel object was successfully created:
##
## Nonlinear mixed-effects model
## Model function: Binary model
## Model type: likelihood
## function(psi,id,xidep) {
##   tim<-xidep[,1]
##   y<-xidep[,2]
##   inter<-psi[id,1]
##   slope<-psi[id,2]
##   logit<-inter+slope*tim
##   pevent<-exp(logit)/(1+exp(logit))
##   pobs = (y==0)*(1-pevent)+(y==1)*pevent
##   logpdf <- log(pobs)
##   return(logpdf)
## }
## Nb of parameters: 2
##   parameter names:  theta1 theta2
##   distribution:
##   Parameter Distribution Estimated
## [1,] theta1    normal      Estimated
## [2,] theta2    normal      Estimated
## Variance-covariance matrix:
##   theta1 theta2
## theta1    1    0
## theta2    0    1
## Covariate model:
##   theta1 theta2
## [1,]    1    1
## Initial values
##   theta1 theta2
## Pop.CondInit  -1.5  -0.1
## Cov.CondInit   0.0   0.0

saemix.options<-list(seed=1234567,save=FALSE,save.graphs=FALSE, displayProgress=FALSE, nb.chains=10, fi

hurdlefit0<-saemix(saemix.hurdle0,saemix.data0,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 rapi.saemix
## Structured data: y0 ~ time + y0 | id
## X variable for graphs: time (week)
## covariates: gender ()

```

```

##      reference class for covariate gender :  Men
## Dataset characteristics:
##      number of subjects:      818
##      number of observations: 3616
##      average/min/max nb obs: 4.42 / 1 / 5
## First 10 lines of data:
##      id time y0 y0.1 gender mdv cens occ ytype
## 1  1  0  1  1  Men  0  0  1  1
## 2  1  6  1  1  Men  0  0  1  1
## 3  1  18  1  1  Men  0  0  1  1
## 4  2  0  0  0  Women  0  0  1  1
## 5  2  6  0  0  Women  0  0  1  1
## 6  2  12  0  0  Women  0  0  1  1
## 7  2  18  0  0  Women  0  0  1  1
## 8  2  24  0  0  Women  0  0  1  1
## 9  3  0  0  0  Men  0  0  1  1
## 10 3  12  0  0  Men  0  0  1  1
## -----
## ----          Model          ----
## -----
## Nonlinear mixed-effects model
## Model function:  Binary model
## Model type:  likelihood
## function(psi,id,xidep) {
##   tim<-xidep[,1]
##   y<-xidep[,2]
##   inter<-psi[id,1]
##   slope<-psi[id,2]
##   logit<-inter+slope*tim
##   pevent<-exp(logit)/(1+exp(logit))
##   pobs = (y==0)*(1-pevent)+(y==1)*pevent
##   logpdf <- log(pobs)
##   return(logpdf)
## }
## <bytecode: 0x55c2785df030>
## Nb of parameters: 2
##   parameter names:  theta1 theta2
##   distribution:
##   Parameter Distribution Estimated
## [1,] theta1    normal      Estimated
## [2,] theta2    normal      Estimated
## Variance-covariance matrix:
##   theta1 theta2
## theta1    1    0
## theta2    0    1
## Covariate model:
##   [,1] [,2]
## gender  1  1
## Initial values
##   theta1 theta2
## Pop.CondInit  -1.5 -0.1
## Cov.CondInit   0.0  0.0
## -----
## ---- Key algorithm options ----

```

```

## -----
## Estimation of individual parameters (MAP)
## Estimation of log-likelihood by importance sampling
## Number of iterations: K1=300, K2=100
## Number of chains: 10
## Seed: 1234567
## 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
## [1,] theta1      -2.796
## [2,] beta_gender(theta1) 0.132
## [3,] theta2       0.036
## [4,] beta_gender(theta2) 0.030
## -----
## ----- Variance of random effects -----
## -----
## Parameter      Estimate
## theta1 omega2.theta1 2.4033
## theta2 omega2.theta2 0.0062
## -----
## ----- Correlation matrix of random effects -----
## -----
##           omega2.theta1 omega2.theta2
## omega2.theta1 1           0
## omega2.theta2 0           1
## -----
## ----- Statistical criteria -----
## -----
## Likelihood computed by importance sampling
##     -2LL= 3249.132
##     AIC = 3263.132
##     BIC = 3296.08
## -----
cat("Expected proportion of 0's at time 0:",1/(1+exp(-hurdlefit0@results@fixed.effects[1])), "\n")

## Expected proportion of 0's at time 0: 0.05753853
table(rapi.saemix$rapi[rapi.saemix$time==0] == 0) # 10.6%

##
## FALSE TRUE
## 731    87

```

```

# Fit Poisson model to saemix.data1
saemix.hurdle1.cov2<-saemixModel(model=count.poisson,description="Count model Poisson",modeltype="likel
                                simulate.function = saemix.simulatePoisson,
                                psi0=matrix(c(log(5),0.01),ncol=2,byrow=TRUE,dimnames=list(NULL, c("in
                                transform.par=c(0,0), omega.init=diag(c(0.5, 0.5)),
                                covariance.model =matrix(data=1, ncol=2, nrow=2),
                                covariate.model=matrix(c(1,1), ncol=2, byrow=TRUE))

##
##
## The following SaemixModel object was successfully created:
##
## Nonlinear mixed-effects model
##   Model function:  Count model Poisson
##   Model type:     likelihood
## function(psi,id,xidep) {
##   time<-xidep[,1]
##   y<-xidep[,2]
##   intercept<-psi[id,1]
##   slope<-psi[id,2]
##   lambda<- exp(intercept + slope*time)
##   logp <- -lambda + y*log(lambda) - log(factorial(y))
##   return(logp)
## }
## <bytecode: 0x55c27c3e0288>
##   Nb of parameters: 2
##       parameter names:  intercept slope
##       distribution:
##       Parameter Distribution Estimated
## [1,] intercept normal      Estimated
## [2,] slope      normal      Estimated
##   Variance-covariance matrix:
##           intercept slope
## intercept          1    1
## slope              1    1
##   Covariate model:
##       intercept slope
## [1,]          1    1
##   Initial values
##           intercept slope
## Pop.CondInit  1.609438  0.01
## Cov.CondInit  0.000000  0.00

saemix.options<-list(seed=632545,save=FALSE,save.graphs=FALSE, displayProgress=FALSE)

hurdlefit1<-saemix(saemix.hurdle1.cov2,saemix.data1,saemix.options)

## Error in solve.default(F0) :
##   Routine Lapack dgesv : le système est exactement singulier : U[2,2] = 0
## Nonlinear mixed-effects model fit by the SAEM algorithm
## -----
## ----          Data          ----
## -----
## Object of class SaemixData

```

```

##      longitudinal data for use with the SAEM algorithm
## Dataset rapi.saemix[rapi.saemix$rapi > 0, ]
##      Structured data: rapi ~ time + rapi | id
##      X variable for graphs: time (week)
##      covariates: gender ()
##      reference class for covariate gender : Men
## Dataset characteristics:
##      number of subjects:      802
##      number of observations: 2860
##      average/min/max nb obs: 3.57 / 1 / 5
## First 10 lines of data:
##      id time rapi rapi.1 gender mdv cens occ ytype
## 4      2    0    3      3 Women  0    0    1    1
## 5      2    6    6      6 Women  0    0    1    1
## 6      2   12    5      5 Women  0    0    1    1
## 7      2   18    4      4 Women  0    0    1    1
## 8      2   24    5      5 Women  0    0    1    1
## 9      3    0    9      9  Men   0    0    1    1
## 10     3   12    1      1  Men   0    0    1    1
## 12     4    0    3      3 Women  0    0    1    1
## 13     4    6    2      2 Women  0    0    1    1
## 14     5    0   35     35 Women  0    0    1    1
## -----
## ----          Model          ----
## -----
## Nonlinear mixed-effects model
## Model function: Count model Poisson
## Model type: likelihood
## function(psi,id,xidep) {
##   time<-xidep[,1]
##   y<-xidep[,2]
##   intercept<-psi[id,1]
##   slope<-psi[id,2]
##   lambda<- exp(intercept + slope*time)
##   logp <- -lambda + y*log(lambda) - log(factorial(y))
##   return(logp)
## }
## <bytecode: 0x55c27c3e0288>
## Nb of parameters: 2
##      parameter names: intercept slope
##      distribution:
##      Parameter Distribution Estimated
## [1,] intercept normal      Estimated
## [2,] slope      normal      Estimated
## Variance-covariance matrix:
##      intercept slope
## intercept      1      1
## slope          1      1
## Covariate model:
##      [,1] [,2]
## gender   1    1
## Initial values
##      intercept slope
## Pop.CondInit 1.609438 0.01

```



```

## Cov.CondInit  0.000000  0.00
## -----
## ----    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(%) p-value
## [1,] intercept              1.8656  0.066   3.5  -
## [2,] beta_gender(intercept) -0.1972  0.089  44.9  0.026
## [3,] slope                  -0.0059  0.057 955.8  -
## [4,] beta_gender(slope)     -0.0085  0.075 881.7  0.910
## -----
## ----- Variance of random effects -----
## -----
##      Parameter              Estimate SE CV(%)
## intercept omega2.intercept    0.6000  NA NA
## slope      omega2.slope        0.0017  NA NA
## covar      cov.intercept.slope -0.0103  NA NA
## -----
## ----- Correlation matrix of random effects -----
## -----
##      omega2.intercept omega2.slope
## omega2.intercept  1.00      -0.32
## omega2.slope     -0.32      1.00
## -----
## ----- Statistical criteria -----
## -----
## Likelihood computed by linearisation
##      -2LL= 437509.5
##      AIC = 437525.5
##      BIC = 437563
##
## Likelihood computed by importance sampling
##      -2LL= 17628.18
##      AIC = 17644.18
##      BIC = 17681.67
## -----

```

```
summary(hurdlefit0)
```

```
## -----
## ----- Fixed effects -----
## -----
##           Parameter Estimate
## 1          theta1    -2.796
## 2 beta_gender(theta1)  0.132
## 3          theta2     0.036
## 4 beta_gender(theta2)  0.030
## -----
## ----- Variance of random effects -----
## -----
##           Parameter Estimate
## theta1 omega2.theta1  2.4033
## theta2 omega2.theta2  0.0062
## -----
## ----- Correlation matrix of random effects -----
## -----
##           omega2.theta1 omega2.theta2
## omega2.theta1 1.00          0.00
## omega2.theta2 0.00          1.00
## -----
## ----- Statistical criteria -----
## -----
##
## Likelihood computed by importance sampling
##      -2LL= 3249.132
##      AIC = 3263.132
##      BIC = 3296.08
## -----
```

```
summary(hurdlefit1)
```

```
## -----
## ----- Fixed effects -----
## -----
## Warning in .local(object, ...): NAs introduits lors de la conversion automatique
##           Parameter Estimate    SE  CV(%) p-value
## 1          intercept    1.8656 0.066   3.53    -
## 2 beta_gender(intercept) -0.1972 0.089  44.92  0.026
## 3              slope   -0.0059 0.057 955.79    -
## 4 beta_gender(slope)   -0.0085 0.075 881.67  0.910
## -----
## ----- Variance of random effects -----
## -----
##           Parameter Estimate SE CV(%)
## intercept omega2.intercept  0.6000 NA   NA
## slope      omega2.slope     0.0017 NA   NA
## -----
## ----- Correlation matrix of random effects -----
## -----
##           omega2.intercept omega2.slope
```

```
## omega2.intercept  1.00          -0.32
## omega2.slope     -0.32          1.00
## -----
## ----- Statistical criteria -----
## -----
## Likelihood computed by linearisation
##      -2LL= 437509.5
##      AIC = 437525.5
##      BIC = 437563
##
## Likelihood computed by importance sampling
##      -2LL= 17628.18
##      AIC = 17644.18
##      BIC = 17681.67
## -----

# Simulate binary data
# proportion of 0's in the data
rapi.tab <- table(rapi.saemix$rapi == 0)

nsim<-100
ysim.hurdle0 <- simulateDiscreteSaemix(hurdlefit0, nsim=nsim)
cat("Observed proportion of 0's overall:",rapi.tab[2]/sum(rapi.tab),"\n")

## Observed proportion of 0's overall: 0.2090708

cat("Simulated proportion of 0's overall:",sum(ysim.hurdle0@sim.data@datasim$ysim)/length(ysim.hurdle0@sim.data@datasim$ysim),"\n")

## Simulated proportion of 0's overall: 0.2069994

# Graph of proportion of 0's with time
yfit<-ysim.hurdle0
simdat <-yfit@sim.data@datasim
simdat$time<-rep(yfit@data@data$time,nsim)
simdat$gender<-rep(yfit@data@data$gender,nsim)

ytab<-NULL
for(irep in 1:nsim) {
  xtab<-simdat[simdat$irep==irep,]
  suppressMessages(
    xtab1 <- xtab %>%
      group_by(time, gender) %>%
      summarise(nev = sum(ysim), n=n()) %>%
      mutate(freq = nev/n)
  )
  ytab<-rbind(ytab,xtab1[,c("time","gender","freq")])
}
gtab <- ytab %>%
  group_by(time, gender) %>%
  summarise(lower=quantile(freq, c(0.05)), median=quantile(freq, c(0.5)), upper=quantile(freq, c(0.95)))
  mutate(gender=ifelse(gender==0,"Men","Women"))

## `summarise()` has grouped output by 'time'. You can override using the
## `.groups` argument.

gtab$freq<-1
gtab3<-cbind(gtab, model="Hurdle")
```

```

gtab<-rbind(gtab1, gtab2, gtab3)
gtab <- gtab %>%
  mutate(model=factor(model, levels=c("Poisson", "ZIP", "Hurdle")))

rapipl <- rapi.saemix %>%
  group_by(time, gender) %>%
  summarise(nev = sum(y0), n=n()) %>%
  mutate(freq = nev/n, sd=sqrt((1-nev/n)/nev)) %>%
  mutate(lower=freq-1.96*sd, upper=freq+1.96*sd)

## `summarise()` has grouped output by 'time'. You can override using the
## `.groups` argument.

rapipl$lower[rapipl$lower<0] <-0 # we should use a better approximation for CI

# Table form - compare to column B in Table 2
yfit0<-hurdlefit0
yfit1<-hurdlefit1

rr.tab<-data.frame(param=c("intercept", "beta.Male.inter", "slope", "beta.Male.slope", "omega.inter", "omega.slope"),
  poissonNoZero=c(yfit1@results@fixed.effects, c(sqrt(diag(yfit1@results@omega)))),
  logistic=c(yfit0@results@fixed.effects, c(sqrt(diag(yfit0@results@omega)))))

print(rr.tab)

##           param poissonNoZero    logistic
## 1      intercept    1.865583452 -2.79604024
## 2 beta.Male.inter   -0.197211376  0.13215067
## 3           slope   -0.005943599  0.03642832
## 4 beta.Male.slope  -0.008525854  0.02950090
## 5      omega.inter    0.774608111  1.55026563
## 6      omega.slope    0.041313987  0.07889691

```

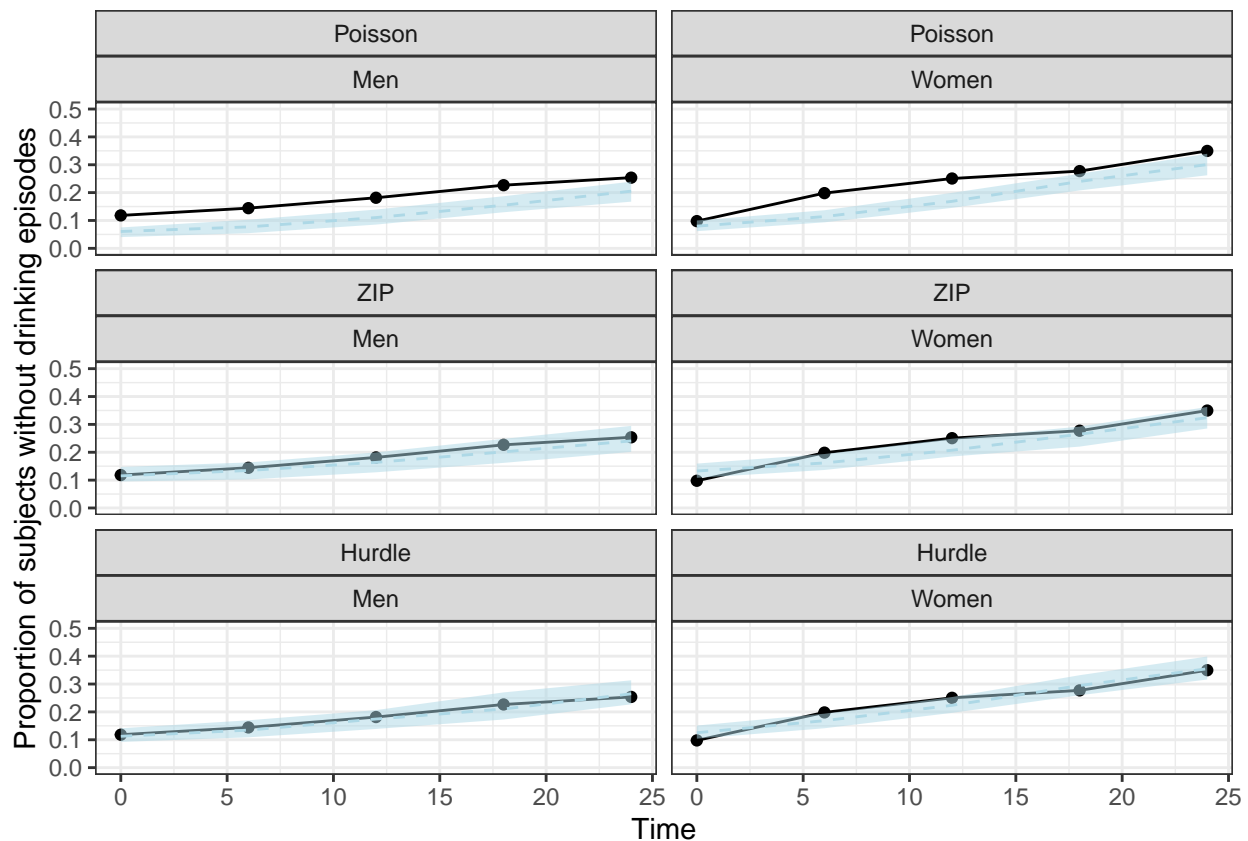
Comparing the proportion of 0's for the different models Clear model misfit for Poisson, much better for the other models with a slight advantage to Hurdle (?).

```

plot2 <- ggplot(rapipl, aes(x=time, y=freq, group=gender)) + geom_line() +
  geom_point() +
  geom_line(data=gtab, aes(x=time, y=median, group=gender), linetype=2, colour='lightblue') +
  geom_ribbon(data=gtab, aes(ymin=lower, ymax=upper, group=gender), alpha=0.5, fill='lightblue') +
  ylim(c(0,0.5)) + theme_bw() + theme(legend.position = "none") + facet_wrap(model~gender, ncol=2) +
  xlab("Time") + ylab("Proportion of subjects without drinking episodes")

print(plot2)

```



```
if(saveForDocs) {
  namfig<-"rapi_comparePropNoDrinking.eps"
  cairo_ps(file = file.path(figDir, namfig), onefile = TRUE, fallback_resolution = 600, height=8.27, width=11.7,
    plot(plot2)
    dev.off()
}
```

Other diagnostics Plot evolution of median score ? VPC on selected scores or on categories (more complicated and not saemix, more using R code with the simulated data)

```
# running fim.saemix to extract the parameters with their name
y1<-fim.saemix(poisson.fit.cov2)
```

Summarising all models in a LaTeX table

```
## Error in solve.default(F0) :
##   Routine Lapack dgesv : le système est exactement singulier : U[2,2] = 0
y2<-fim.saemix(zipppoisson.fit.cov2)
```

```
## Error in solve.default(F0) :
##   Routine Lapack dgesv : le système est exactement singulier : U[2,2] = 0
y3<-fim.saemix(hurdlefit1)
```

```
## Error in solve.default(F0) :
##   Routine Lapack dgesv : le système est exactement singulier : U[2,2] = 0
```

```

parnam<-as.character(y2@results@conf.int$name)
estpar<-data.frame(parameter=parnam, poisson=y1@results@conf.int$estimate[match(parnam,y1@results@conf.int$parameter)],
  zip=y2@results@conf.int$estimate[match(parnam,y2@results@conf.int$parameter)],
  hurdle=y3@results@conf.int$estimate[match(parnam,y3@results@conf.int$parameter)])
estpar[estpar$parnam=="p0", 4]<-hurdlefit0@results@fixed.effects[3]
estpar<-estpar[-c(6:7),]
for(icol in 2:4)
  estpar[,icol]<-format(estpar[,icol], digits=1, ns=1)

print(estpar)

```

```

##           parameter poisson  zip hurdle
## 1           intercept    1.68  1.77  1.866
## 2 beta_gender(intercept) -0.20 -0.20 -0.197
## 3                slope  -0.02 -0.02 -0.006
## 4   beta_gender(slope)  -0.02 -0.02 -0.009
## 5                p0      NA   0.08    NA
## 8          SD.intercept   0.96  0.88  0.775
## 9          SD.slope     0.06  0.06  0.041

```

Time-to-event

TTE model - lung cancer

- create dataset for saemix (once) using the lung data from the survival package [see examplesDocumentation.R]
- changes
 - saemix format: added time=0
 - created one column for status (dead or alive, recoded as 1/0) and one for censoring (0/1)
 - removed subjects with missing age, institution, sex, or ph scores (ecog and karno)

Checks

- The `Surv` function from the `survival` package creates a survival object for use as the response in a model formula.
 - one entry for each subject that is the survival time, which is followed by a `+` if the subject was censored
 - transform `lung.saemix` in the `Surv` format to check the survival function w/r saemix fit
- Weibull model
 - parametric survival function

$$S(t) = e^{-\left(\frac{t}{\lambda}\right)^\beta}$$

- Also tried computing a SE for $S(t)$ using the delta-method where the vector of derivatives for the survival function of Weibull model are:

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

- works pretty well compared to the non-parametric KM estimate

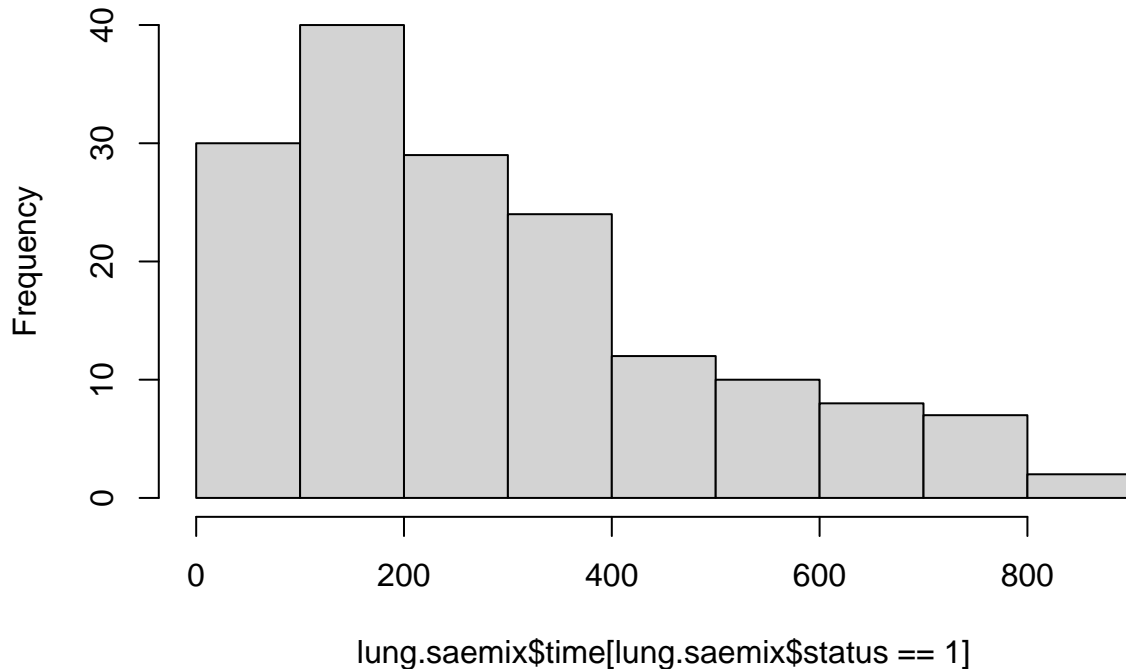
```

if(testMode)
  data(lung.saemix) else
  lung.saemix<-read.table(file.path(datDir, "lung.saemix.tab"), header=TRUE)

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

```

Histogram of lung.saemix\$time[lung.saemix\$status == 1]



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

saemix.data<-saemixData(name.data=lung.saemix,header=TRUE,name.group=c("id"),
  name.predictors=c("time","status","cens"),name.response=c("status"),
  name.covariates=c("age","sex","ph.ecog","ph.karno","pat.karno","wt.loss","meal.cal"),
  units=list(x="days",y="",covariates=c("yr","","-","%","%","cal","pounds")))

##
##
## The following SaemixData object was successfully created:
##
## Object of class SaemixData
##   longitudinal data for use with the SAEM algorithm
## Dataset lung.saemix
##   Structured data: status ~ time + status + cens | id
##   X variable for graphs: time (days)
##   covariates: age (yr), sex (), ph.ecog (-), ph.karno (%), pat.karno (%), wt.loss (cal), meal.cal
##   reference class for covariate sex : 0

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)
  lambda <- psi[id,1] # Parameters of the Weibull model
  beta <- psi[id,2]
  Nj <- length(T)
```

```

ind <- setdiff(1:Nj, append(init,cens)) # indices of events
hazard <- (beta/lambda)*(T/lambda)^(beta-1) # ln(H')
H <- (T/lambda)^beta # ln(H)
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<-saemixModel(model=weibulltte.model,description="time model",modeltype="likelihood",
  psi0=matrix(c(1,2),ncol=2,byrow=TRUE,dimnames=list(NULL, c("lambda","beta"))),
  transform.par=c(1,1),covariance.model=matrix(c(1,0,0,0),ncol=2, byrow=TRUE))

##
##
## The following SaemixModel object was successfully created:
##
## Nonlinear mixed-effects model
## Model function: time 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)
##   lambda <- psi[id,1] # Parameters of the Weibull model
##   beta <- psi[id,2]
##   Nj <- length(T)
##
##   ind <- setdiff(1:Nj, append(init,cens)) # indices of events
##   hazard <- (beta/lambda)*(T/lambda)^(beta-1) # ln(H')
##   H <- (T/lambda)^beta # ln(H)
##   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)
## }
## Nb of parameters: 2
##   parameter names: lambda beta
##   distribution:
##   Parameter Distribution Estimated
## [1,] lambda    log-normal    Estimated
## [2,] beta      log-normal    Estimated
## Variance-covariance matrix:
##   lambda beta
## lambda      1    0
## beta        0    0
## No covariate in the model.
## Initial values
##   lambda beta
## Pop.CondInit      1    2

saemix.options<-list(seed=632545,save=FALSE,save.graphs=FALSE, displayProgress=FALSE)
tte.fit<-saemix(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 lung.saemix
##   Structured data: status ~ time + status + cens | id
##   X variable for graphs: time (days)
##   covariates: age (yr), sex (), ph.ecog (-), ph.karno (%), pat.karno (%), wt.loss (cal), meal.cal
##   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 age sex ph.ecog ph.karno pat.karno wt.loss
## 1  1  0      0  0      0  74  0      1      90      100      NA
## 2  1 306      1  0      1  74  0      1      90      100      NA
## 3  2  0      0  0      0  68  0      0      90      90      15
## 4  2 455      1  0      1  68  0      0      90      90      15
## 5  3  0      0  0      0  56  0      0      90      90      15
## 6  3 1010     0  1      0  56  0      0      90      90      15
## 7  4  0      0  0      0  57  0      1      90      60      11
## 8  4 210      1  0      1  57  0      1      90      60      11
## 9  5  0      0  0      0  60  0      0     100      90      0
## 10 5 883      1  0      1  60  0      0     100      90      0
##   meal.cal mdv cens.1 occ ytype
## 1      1175  0      0  1      1
## 2      1175  0      0  1      1
## 3      1225  0      0  1      1
## 4      1225  0      0  1      1
## 5         NA  0      0  1      1
## 6         NA  0      0  1      1
## 7      1150  0      0  1      1
## 8      1150  0      0  1      1
## 9         NA  0      0  1      1
## 10        NA  0      0  1      1
## -----
## ----          Model          ----
## -----
## Nonlinear mixed-effects model
##   Model function: time 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)
##   lambda <- psi[id,1] # Parameters of the Weibull model
##   beta <- psi[id,2]
##   Nj <- length(T)
##
##   ind <- setdiff(1:Nj, append(init,cens)) # indices of events

```

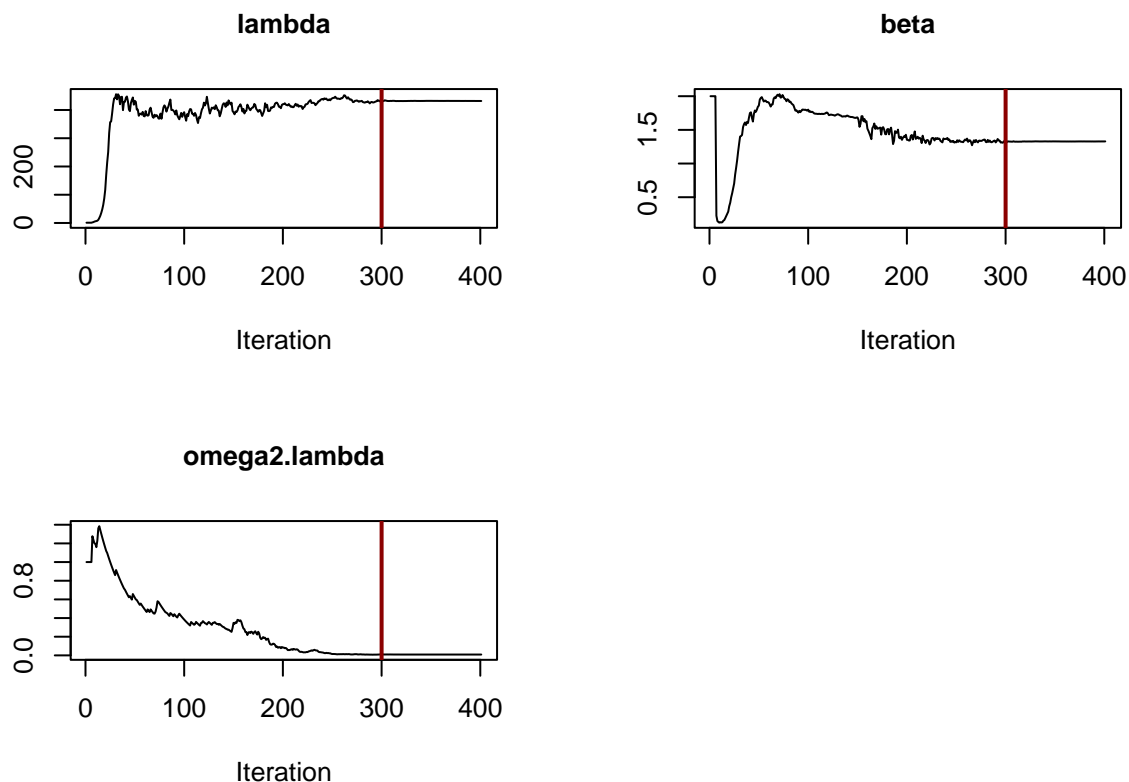
```

## hazard <- (beta/lambda)*(T/lambda)^(beta-1) # ln(H')
## H <- (T/lambda)^beta # ln(H)
## 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: 0x55c278f6b9d8>
## Nb of parameters: 2
## parameter names: lambda beta
## distribution:
## Parameter Distribution Estimated
## [1,] lambda log-normal Estimated
## [2,] beta log-normal Estimated
## Variance-covariance matrix:
## lambda beta
## lambda 1 0
## beta 0 0
## No covariate in the model.
## Initial values
## lambda beta
## 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,] lambda 431.8 51.60 12
## [2,] beta 1.3 0.19 14
## -----
## ----- Variance of random effects -----
## -----
## Parameter Estimate SE CV(%)
## lambda omega2.lambda 0.009 0.17 1858
## -----
## ----- Correlation matrix of random effects -----
## -----

```

```
##               omega2.lambda
## omega2.lambda 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
## -----
```

```
plot(tte.fit, plot.type="convergence")
```



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

# Use survival package to assess Survival curve
if(TRUE) {
  library(survival)
  lung.surv<-lung.saemix[lung.saemix$time>0,]
  lung.surv$status<-lung.surv$status+1
  Surv(lung.surv$time, lung.surv$status) # 1=censored, 2=dead
  f1 <- survfit(Surv(time, status) ~ 1, data = lung.surv)
  xtim<-seq(0,max(lung.saemix$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]
dsdbeta<- -log(xtim/estpar[1]) * xcal *exp(-xcal)
dsdalpha<- estpar[2]/estpar[1] * xcal *exp(-xcal)
xmat<-rbind(dsdalpha, dsdbeta)
# 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])
if(saveForDocs) {
  namfile<-"lung_compareKM.eps"
  postscript(file.path(figDir, namfile), horizontal=TRUE)
  plot(f1, 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)
  dev.off()
}

# ypred2<-exp(-(xtim/(estpar[1]-1.96*sqrt(invfim[1,1]))^(estpar[2]+1.96*sqrt(invfim[2,2]))))
# ypred3<-exp(-(xtim/(estpar[1]+1.96*sqrt(invfim[1,1]))^(estpar[2]+1.96*sqrt(invfim[2,2]))))
# lines(xtim,ypred2, col="blue",lwd=1, lty=2)
# lines(xtim,ypred3, col="blue",lwd=1, lty=2)
}

```

RTTE model

- again difficult to find real data
- simulated data
 - Exemple simulé de Belhal **TODO**
 - data from the Monolix documentation: absolutely no indication where the data comes from (weibull_data.txt for the weibullRTTE.mlxtran project in the demo)
- search for real data
 - asked Ulrika Simonsson for the RTTE data on post-operative pain (Pain Medicine 2015)
 - data on events in Gaucher disease used for the ENSAI workshops (but few events)
 - discretised PCA events during warfarin treatment ? (from the warfarin PK/PD) (but threshold ?)

```

# 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
  lambda <- psi[,1]
  beta <- psi[,2]
  simdat<-NULL
  N<-nrow(psi)
  for(i in 1:N) {
    eventTimes<-c(0)
    T<-0
    Vj<-runif(1)
    # T <- (-log(Vj)*lambda[i])^(beta[i])
    T<-lambda[i]*(-log(Vj))^(1/beta[i])
  }
}

```

```

nev<-0
while (T < censoringtime & nev<maxevents){
  eventTimes <- c(eventTimes, T)
  nev<-nev+1
  Vj<-runif(1)
  #      T <- T+(-log(Vj)*lambda[i])^(beta[i])
  #      T<-(-log(Vj)*lambda[i] + T^(1/beta[i]))^(beta[i])
  T<-lambda[i]*(-log(Vj) + (T/lambda[i])^(beta[i]))^(1/beta[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(lambda=param[1]*exp(rnorm(nsuj,sd=omega[1])), beta=param[2]*exp(param[3]*risk+rnorm(nsuj,sd=omega[2])),
simdat <- simul.rtte.unif(psiM)

## Reached maximum number of events
simdat$risk<-as.integer(simdat$id>(nsuj/2))

# Simulate T from Weibull (check)
if(FALSE) {
  lambda<-2
  beta<-2
  nsim<-5000
  # By hand
  q1<-runif(nsim)
  # tevent<-lambda*exp(log(q1)/beta)
  tevent<-lambda*exp(log(-log(q1))/beta)
  tevent<-sort(tevent)
  # plot(tevent, exp(-(tevent/lambda)^beta))
  tevent2<-sort(rweibull(nsim, shape=beta, scale=lambda))
  plot(tevent, tevent2)
  abline(0,1)
}

```

```

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

##
##
## The following SaemixData object was successfully created:
##
## Object of class SaemixData
##      longitudinal data for use with the SAEM algorithm
## Dataset simdat
##      Structured data: status ~ T | id
##      Predictor: T ()
##      covariates: risk (-)
##      reference class for covariate risk : 0

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
  lambda <- psi[id,1]
  beta <- 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 <- (beta/lambda)*(T/lambda)^(beta-1)
  H <- (T/lambda)^beta
  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("lambda","be
                                transform.par=c(1,1),covariance.model=matrix(c(1,0,0,1),ncol=2, byrow=TR

##
##
## The following SaemixModel object was successfully created:
##
## Nonlinear mixed-effects model
##      Model function: Repeated TTE model
##      Model type: likelihood
## 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
##      lambda <- psi[id,1]
##      beta <- 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 <- (beta/lambda)*(T/lambda)^(beta-1)
##   H <- (T/lambda)^beta
##   logpdf <- rep(0,Nj)
##   logpdf[tcens] <- -H[tcens] + H[tcens-1]
##   logpdf[tevent] <- -H[tevent] + H[tevent-1] + log(hazard[tevent])
##   return(logpdf)
## }
##   Nb of parameters: 2
##       parameter names:  lambda beta
##       distribution:
##       Parameter Distribution Estimated
## [1,] lambda    log-normal    Estimated
## [2,] beta      log-normal    Estimated
##   Variance-covariance matrix:
##       lambda beta
## lambda      1    0
## beta        0    1
##   No covariate in the model.
##   Initial values
##       lambda beta
## Pop.CondInit      1    2

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("lambda","beta")),
                           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))

##
##
## The following SaemixModel object was successfully created:
##
## Nonlinear mixed-effects model
##   Model function:  Repeated TTE model
##   Model type:     likelihood
## 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
##   lambda <- psi[id,1]
##   beta <- 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 <- (beta/lambda)*(T/lambda)^(beta-1)
##   H <- (T/lambda)^beta
##   logpdf <- rep(0,Nj)
##   logpdf[tcens] <- -H[tcens] + H[tcens-1]
##   logpdf[tevent] <- -H[tevent] + H[tevent-1] + log(hazard[tevent])
##   return(logpdf)
## }
##   Nb of parameters: 2
##       parameter names:  lambda beta

```

```

##      distribution:
##      Parameter Distribution Estimated
## [1,] lambda    log-normal    Estimated
## [2,] beta      log-normal    Estimated
##      Variance-covariance matrix:
##      lambda beta
## lambda      1    0
## beta        0    1
##      Covariate model:
##      lambda beta
## [1,]      0    1
##      Initial values
##      lambda beta
## Pop.CondInit      1    2
## Cov.CondInit      0    0

saemix.options<-list(seed=632545,save=FALSE,save.graphs=FALSE, fim=FALSE, displayProgress=FALSE)
rtte.fit<-saemix(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 simdat
##      Structured data: status ~ T | id
##      Predictor: T ()
##      covariates: risk (-)
##      reference class for covariate risk : 0
## Dataset characteristics:
##      number of subjects:      200
##      number of observations: 967
##      average/min/max nb obs: 4.83 / 2 / 32
## First 10 lines of data:
##      id      T status risk mdv cens occ ytype
## 1  1 0.0000000      0    0  0    0  1      1
## 2  1 0.7520145      1    0  0    0  1      1
## 3  1 0.8775847      1    0  0    0  1      1
## 4  1 2.4331650      1    0  0    0  1      1
## 5  1 3.0000000      0    0  0    0  1      1
## 6  2 0.0000000      0    0  0    0  1      1
## 7  2 1.3712351      1    0  0    0  1      1
## 8  2 3.0000000      0    0  0    0  1      1
## 9  3 0.0000000      0    0  0    0  1      1
## 10 3 2.8564910      1    0  0    0  1      1
## -----
## ----          Model          ----
## -----
## Nonlinear mixed-effects model
##      Model function: Repeated TTE model
##      Model type: likelihood
## function(psi,id,xidep) {
##      T<-xidep[,1]
##      N <- nrow(psi) # nb of subjects

```



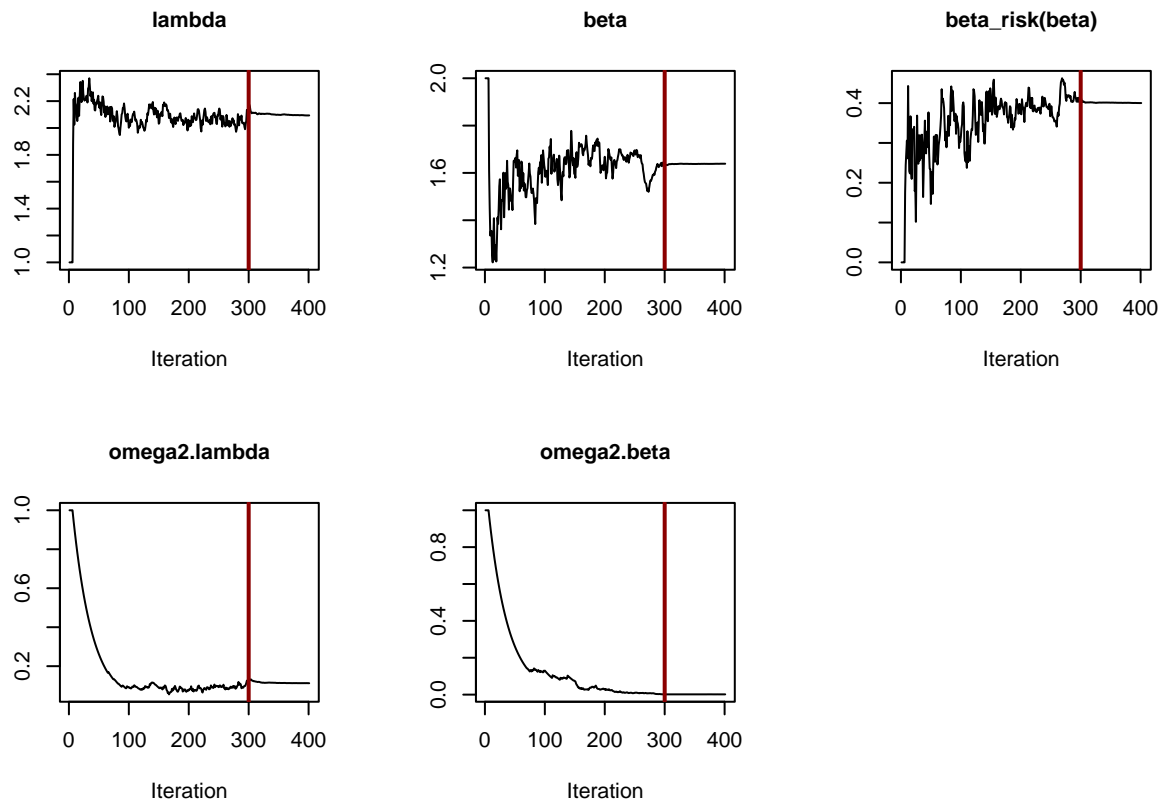
```

## Nj <- length(T) # nb of events (including 0 and censoring times)
## # censoringtime = 6
## censoringtime = max(T) # same censoring for everyone
## lambda <- psi[id,1]
## beta <- 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 <- (beta/lambda)*(T/lambda)^(beta-1)
## H <- (T/lambda)^beta
## logpdf <- rep(0,Nj)
## logpdf[tcens] <- -H[tcens] + H[tcens-1]
## logpdf[tevent] <- -H[tevent] + H[tevent-1] + log(hazard[tevent])
## return(logpdf)
## }
## <bytecode: 0x55c286b9d418>
## Nb of parameters: 2
##     parameter names:  lambda beta
##     distribution:
##     Parameter Distribution Estimated
## [1,] lambda    log-normal  Estimated
## [2,] beta      log-normal  Estimated
## Variance-covariance matrix:
##     lambda beta
## lambda      1    0
## beta        0    1
## Covariate model:
##     [,1] [,2]
## risk    0    1
## Initial values
##     lambda beta
## Pop.CondInit      1    2
## Cov.CondInit      0    0
## -----
## ---- Key algorithm options ----
## -----
## Estimation of individual parameters (MAP)
## 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

```

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

```
plot(rtte.fit, plot.type="convergence")
```



Exiting

```
if(testMode) {  
    dev_mode()  
}
```

```
## v Dev mode: OFF
```