```
   transform.par=c(1))
saemix.options<-list(seed=632545,save=FALSE,save.graphs=FALSE,
   displayProgress=FALSE)
poisson.fit<-saemix(saemix.model,saemix.data,saemix.options)
```

RAPI data

This dataset was kindly made available by David Atkins (University of Washington) in his tutorial on modelling count data [1]. The RAPI data studies gender differences across two years in alcohol-related problems, as measured by the Rutgers Alcohol Problem Index (RAPI) [44]. Students were asked to report every six months the number of alcohol-related problems, and the dataset includes 3,616 repeated measures of these counts in 818 subjects, 561 of whom had the full 5 measurements over a period of 2 years. Interesting features of this dataset are first, the longitudinal aspect which allow to evaluate changes over time, and second, the shape of the distribution of counts. Counts are often positively skewed, bounded by zero, with a large stack of data points at zero, indicating individuals and/or occasions without drinking, use, or related problems. This dataset was used in [1] to illustrate mixed effects count regression using the glmer() function from the lme4.

The dataset is available in saemix under the name rapi.saemix so we read it and create our saemixData object in the usual way. Because we need the value of the outcome to compute the corresponding likelihood, the rapi column is used both as a predictor and as the response:

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

saemix currently does not have automated plots for discrete outcome data, but we can produce our own histogram (here, across all measurements without taking time into account) to notice that indeed, there seems to be many subjects reporting no alcohol related problems over some periods.

Poisson model: the first model we can fit to this data is, as previously, the Poisson model, but this time we add a time effect. Here we will write the same model as in glmer() to compare our results. In glmer() a logarithmic link function is used to transform the mean of the Poisson model (λ) into a linear predictor of time and covariates. Random effects are then added to the parameters of the linear model. To take into account the change with time in saemix, we need to rewrite the previous model to use normal distributions for the parameters and explicitely write the linear model in the function, as follows:

```
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)
}
```

The expression of logp in the model function is unchanged, but now we define a log-normal distribution for λ within the model so that we can use two parameters and time as a predictor. The statistical model also changes to reflect this, as our parameters intercept and slope are now on the scale of the random effects, so they are given a normal distribution. Defining and fitting this model in saemix, we have:

```
saemix.model.poi<-saemixModel(model=count.poisson,description="Count model Poisson",
    modeltype="likelihood",
    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)))
saemix.options<-list(seed=632545,save=FALSE,save.graphs=FALSE, displayProgress=FALSE)
poisson.fit<-saemix(saemix.model.poi,saemix.data,saemix.options)
```

Note that when parameters enter the model through a normal distribution, we may need to adjust the initial values of the Ω matrix (argument omega.init) to avoid failure to find valid initial parameter estimates.

We can also add the covariate gender to both parameters as well as a correlation between the two random effects:

```
modsmx.poi.cov2<-saemixModel(model=count.poisson,
      description="Count model Poisson",modeltype="likelihood",
      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)),
      covariance.model=matrix(data=1, ncol=2, nrow=2),
      covariate.model=matrix(c(1,1), ncol=2, byrow=TRUE))
poisson.fit.cov2<-saemix(modsmx.poi.cov2,saemix.data,saemix.options)
```

Comparing the parameter estimates from this fit to the estimates obtained by glmer() using a Laplace approximation in Table 2 of [1], we find very good agreement with the SAEM algorithm.

Note: saemix does not provide adequate standard errors of estimation for the parameters in version 3.0. The FO-approximation of the FIM implemented in the current version of the algorithm is known to be very poor for discrete outcome models.

Some diagnostics for this model can be obtained by simulating from the model. To to this we need to define a simulation function associated with the structural model, with the same arguments as the model function, and returning simulated responses. For the Poisson model, this function would be the following, where we replace the line defining the log-probability logp in count.poisson with a line simulating from a Poisson distribution with parameter λ for each value of time, and returning those simulated counts.

```
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)
}
```

We then use the simulateDiscreteSaemix function to obtain simulations from the model, using the estimated parameters. Here we set the number of simulations to 100 as the dataset is large and we are interested in global diagnostics.

```
yfit1<-simulateDiscreteSaemix(poisson.fit.cov2, simulate.function=saemix.simulatePoisson,
    nsim=100)
cat("Observed proportion of 0's",
     length(yfit1@data@data$rapi[yfit1@data@data$rapi==0])/yfit1@data@ntot.obs,"\n")
cat("      Poisson model, p=",
    length(yfit1@sim.data@datasim$ysim[yfit1@sim.data@datasim$ysim==0])/
    length(yfit1@sim.data@datasim$ysim),"\n")
```

Handling overdispersion: the model predicts a lower proportion of subjects without alcohol-related problems than we observe in data, a sign of overdispersion (with a Poisson model, the mean of the Poisson distribution, λ, is equal to the variance, an assumption which is violated here). Several models can be used to take this feature into account. First, we can use the Zero-Inflated Poisson model, where the number of counts equal to 0 is increased. This model can be built as a mixture between a distribution of 0's with probability p_0 and a standard Poisson model. We modify our model function above to:

```
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)
}
```

and fit the model using:

```
modsmx.zip<-saemixModel(model=count.poissonzip,description="count model ZIP",
   modeltype="likelihood",
   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=diag(c(0.5,0.3,0)),
   covariate.model = matrix(c(1,1,0),ncol=3, byrow=TRUE))

zippoisson.fit <- saemix(modsmx.zip,saemix.data,saemix.options)
```

where we assume a logit-normal distribution for the added parameter p0 through the transform.par argument. We could also model logit(p0) on a normal scale if we needed to add a time effect. The same approach as above can then be used to simulate from the model, this time using the simulation function:

```
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)
}
```

and we can check that the proportion of simulated 0's is now closer to the observed value.

A second type of model used in [1] is the hurdle model, which combines a binary logistic model for the probability of having counts greater than 0, with a truncated Poisson model for counts greater than 0. To implement this, we fit separately the two models: the binary logistic model is fit to the data where we use as a response the binary outcome with values 0 for counts of 0 and 1 for counts strictly positive, then the Poisson model is fit to the data excluding zero counts.

Other possible models include the negative binomial model and generalised Poisson models with additional parameters handling the overdispersion.

Diagnostics: the simulation functions can be used to produce diagnostic plots. As an example, we can compare the expected proportion of 0's, representing subjects without alcohol problems, versus time and stratified by gender, to compare the different models (the code available as a notebook on the github for saemix). The results, shown in Figure 4.24. We could also look at the counts for different categories or the evolution of the median counts.

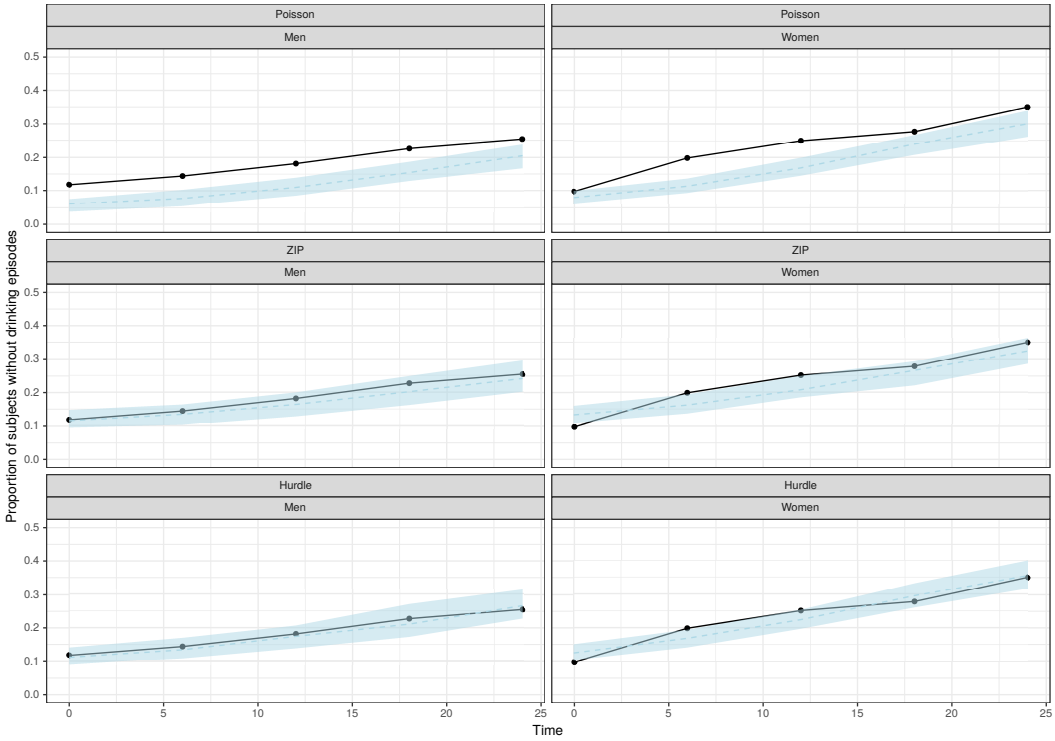

Figure 4.24: Proportion of subjects without drinking problems versus time, for men and women, observed and expected for the Poisson, ZIP and hurdle models, for the RAPI data.

4.7 Time-to-event data

4.7.1 Single event

The example chosen to illustrate the analysis of time-to-event data is the NCCTG Lung Cancer Data, describing the survival in patients with advanced lung cancer from the North Central Cancer Treatment Group [24]. Covariates measured in the study include performance scores rating how well the patient can perform usual daily activities. We reformatted the cancer dataset provided in the survival package in R [41] in SAEM format: patients with missing age, sex, institution or physician assessments were removed from the dataset. Status was recoded as 1 for death and 0 for a censored event, and a censoring column was added to denote whether the patient was dead or alive at the time of the last observation. A line at time=0 was added for all subjects. Finally, subjects were numbered consecutively from 0 to 1.

We can use a Weibull model for the hazard, parameterised as λ and β. For individual i, the hazard function of this model is:

$$h(t,\psi_i) = \frac{\beta_i}{\lambda_i}\left(\frac{t}{\lambda_i}\right)^{\beta_i-1} . \tag{4.12}$$

Here, the vector of individual parameters is $\psi_i = (\lambda_i, \beta_i)$. These two parameters are assumed to be independent and lognormally distributed:

$$\log(\lambda_i) \sim \mathcal{N}(\log(\lambda_{\text{pop}}), \omega_\lambda^2) , \tag{4.13}$$
$$\log(\beta_i) \sim \mathcal{N}(\log(\beta_{\text{pop}}), \omega_\beta^2) . \tag{4.14}$$

Then, the vector of population parameters is $\theta = (\lambda_{\text{pop}}, \beta_{\text{pop}}, \omega_\lambda, \omega_\beta)$.

The survival function for this model is:

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

The model function for saemix needs to define the log-pdf for each observation. At time 0, it is 0 (no event has occurred yet). For a censored event, the log-likelihood is equal to the logarithm of the survival function since the beginning of the observation period, while for an observed event we add the logarithm of the hazard at the time of the event. In the model below, we pass individual censoring times as the third predictor, so that each individual may have his or her own follow-up duration.

```
weibulltte.model<-function(psi,id,xidep) {
  T<-xidep[,1]
  y<-xidep[,2] # events (1=event, 0=no event)
```