Saemix 3 - count data models

Emmanuelle

08/2022

Version

Use saemix version $\geq$ 3.2

Objective

Run binary and categorical models in **saemix**

This notebook uses additional code from the **saemix** development github (https://github.com/saemixdevelopment/saemixextension), not yet integrated in the package. The *workDir* folder in the next chunk of code points to the folder where the user stored this code, and is needed to run the notebook (*workDir* defaults to the current working directory). Specifically, the notebook loads:

- code for the MC/AGQ provided by Sebastian Ueckert (Ueckert et al. 2017)
  - if memory issues arise the code can be run in a separate script.
- the results for the bootstrap runs performed using different approaches (see Comets et al. Pharm Res 2021)
  - bootstraps can be run instead by switching the *runBootstrap* variable to TRUE in the first chunk of code
  - in the code, the number of bootstraps is set to 10 for speed but we recommend to use at least 200 for a 90% CI.
  - this can be changed in the following change of code by uncommenting the line *nboot<-200* and setting the number of bootstrap samples (this may cause memory issues in **Rstudio** with older machines, if this is the case we recommend executing the code in a separate script)

The current notebook can be executed to create an HMTL or PDF output with comments and explanations. A script version containing only the R code is also given as *saemix3_countModel.R* in the same folder.

Count data model

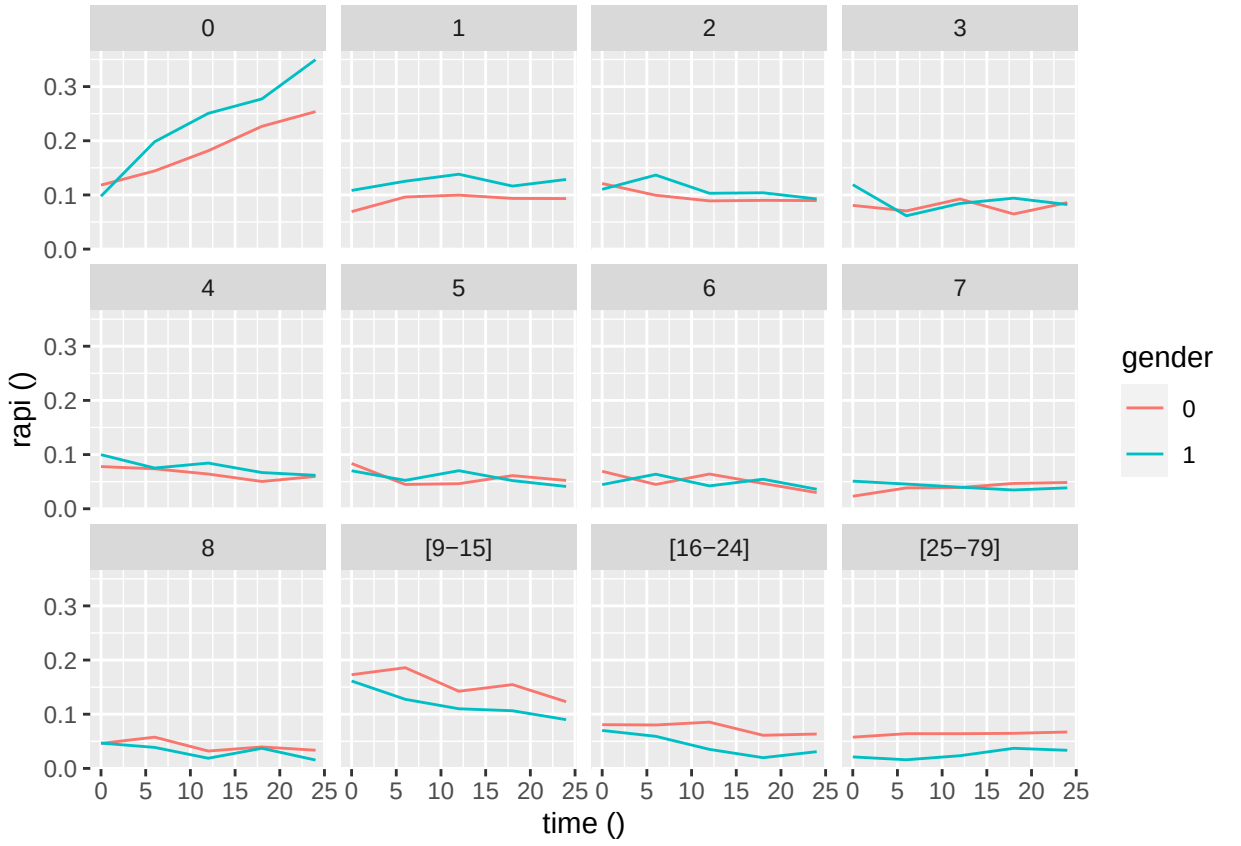
Data description The *rapi.saemix* dataset in the **saemix** package contains count data kindly made available by David Atkins (University of Washington) in his tutorial on modelling count data (Atkins et al. 2013). The data comes from a randomised controlled trial assessing the effectiveness of web-based personalised normative feedback intervention on alcohol consumption (Neighbors et al. 2010a, 2010b}. The *rapi.saemix* dataset records alcohol-related problems, as measured by the Rutgers Alcohol Problem Index (RAPI) (White et al. 1989), in freshmen at risk for heavy drinking behaviours. 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 Atkins et al. (2013) to illustrate mixed effects count regression using the *glmer()* function from the **lme4** package.

Similarly to categorical data, we need the value of the outcome to compute the associated likelihood. Therefore, to create the data object using saemixData, we need to specify the response column both as a response (*name.response="rapi"*) and as a predictor (here, time is the first predictor and we add the response in the argument *name.predictors*).

```
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("")), verbose=FALSE)
```

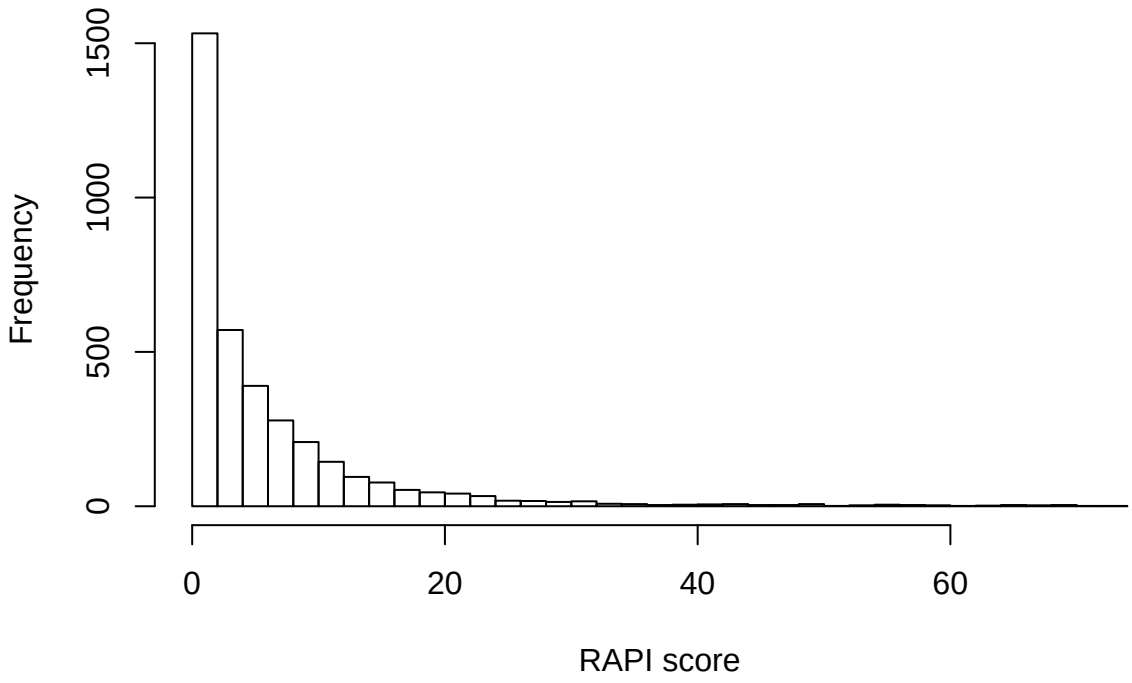
Exploring data We can visualise the evolution of the proportion of each count over time using the *plotDiscreteData()* function with the outcome set to *count*. Here we stratify by gender and regroup the higher scores in 3 categories.

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

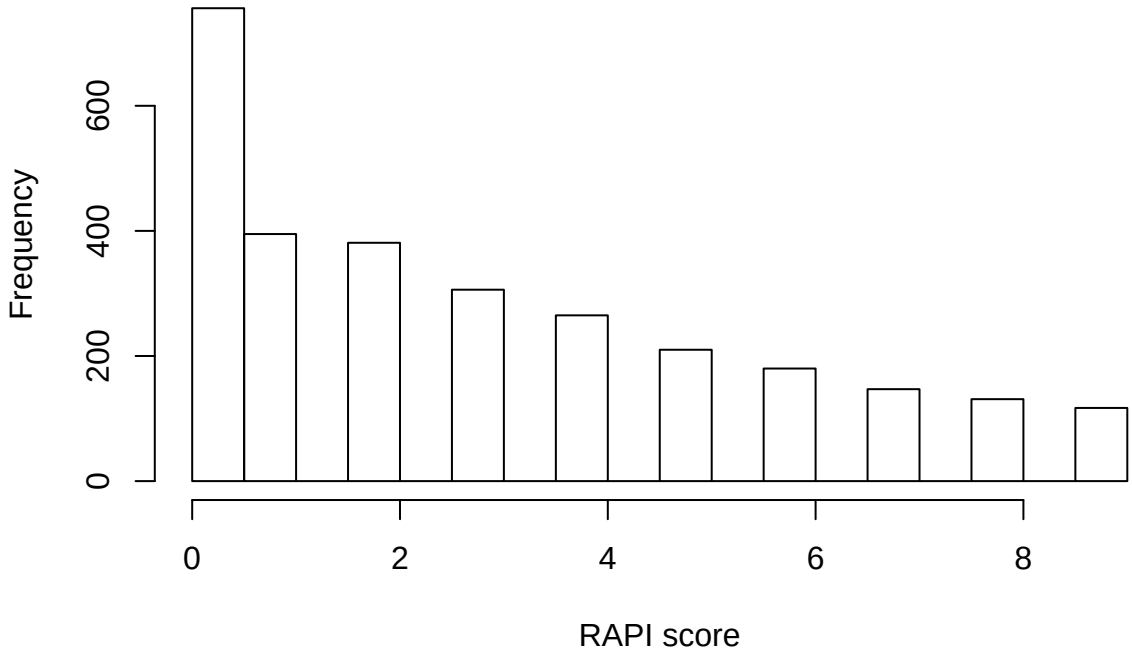


The distribution of count data can also be visualised as a histogram. The over-representation of low scores can be seen when zooming on the early part of the histogram. We can also tabulate the data by stratifying on men and women to realise that there seems to be a gender difference, with more women not reporting any episode of drinking than men.

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




```
# Zooming on small values of scores
hist(rapi.saemix$rapi[rapi.saemix$rapi < 10], main="", xlab="RAPI score", breaks=30)
```




```
table(rapi.saemix$gender, as.integer(rapi.saemix$rapi > 2))
```

```
##
##             0    1
##   Men     548  938
##   Women   984 1146
```

Statistical model Several models can be fit to the data.

The simplest one is a Poisson model which assumes that the probability to observe a count value equal to n is given by:

$$P(Y = n) = \frac{\lambda^n \, e^{-\lambda}}{n!} \tag{1}$$

where $\lambda > 0$ is the parameter of the model. We assume a time effect λ, which is defined as a linear function of time.

For the statistical model, we assume a normal distribution for intercept and slope, so that the distribution of λ is log-normal.

For the simulation function, we add a test to detect outlier values (above the maximum observed value in the original dataset) - not doing this leads to unrealistic simulated values orders of magnitude larger than the maximum observed value, and causes in particular the residual bootstraps to fail - disclaimer: this is a workaround and may not be the proper statistical solution ! (if you have an idea about how to avoid aberrant values when simulating from a Poisson model, feel free to drop me a line at emmanuelle.comets@inserm.fr)

```
## Poisson with a time effect
# Model
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)
}
# Simulation function
countsimulate.poisson<-function(psi, id, xidep) {
  time<-xidep[,1]
  y<-xidep[,2]
  ymax<-max(y)
  intercept<-psi[id,1]
  slope<-psi[id,2]
  lambda<- exp(intercept + slope*time)
  y<-rpois(length(time), lambda=lambda)
  y[y>ymax]<-ymax+1 # truncate to maximum observed value to avoid simulating aberrant values
  return(y)
}
```

Fitting models

Poisson model The code below fits a base Poisson model without correlation between the parameters, as well as a model with a covariance between the two parameters and a gender effect on both parameters.

```
## Poisson
### Model without covariate
saemix.model.poi<-saemixModel(model=count.poisson,description="Count model Poisson",simulate.function=c
                              modeltype="likelihood",
                              psi0=matrix(c(log(5),0.01),ncol=2,byrow=TRUE,dimnames=list(NULL, c("inter
                              transform.par=c(0,0), omega.init=diag(c(0.5, 0.5)), verbose=FALSE)

### Gender effect on intercept and slope
saemix.model.poi.cov2<-saemixModel(model=count.poisson,description="Count model Poisson",simulate.funct
                                   modeltype="likelihood",
```

```
                                    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), verbose=FALSE)

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

### Fit with saemix
poisson.fit<-saemix(saemix.model.poi,saemix.data,saemix.options)
poisson.fit.cov2<-saemix(saemix.model.poi.cov2,saemix.data,saemix.options)
#print(poisson.fit@results)
summary(poisson.fit.cov2)
```

```
## ----------------------------------------------------
## -----------------  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
## ----------------------------------------------------
```

```
### Results
if(FALSE) {
  cat("Poisson parameter at time 0 in base model: lambda_0=", exp(poisson.fit@results@fixed.effects[1])
  cat("Poisson parameter at time 24 in base model: lambda_24=", exp(poisson.fit@results@fixed.effects[1]
}

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

- Results
  - numerical output

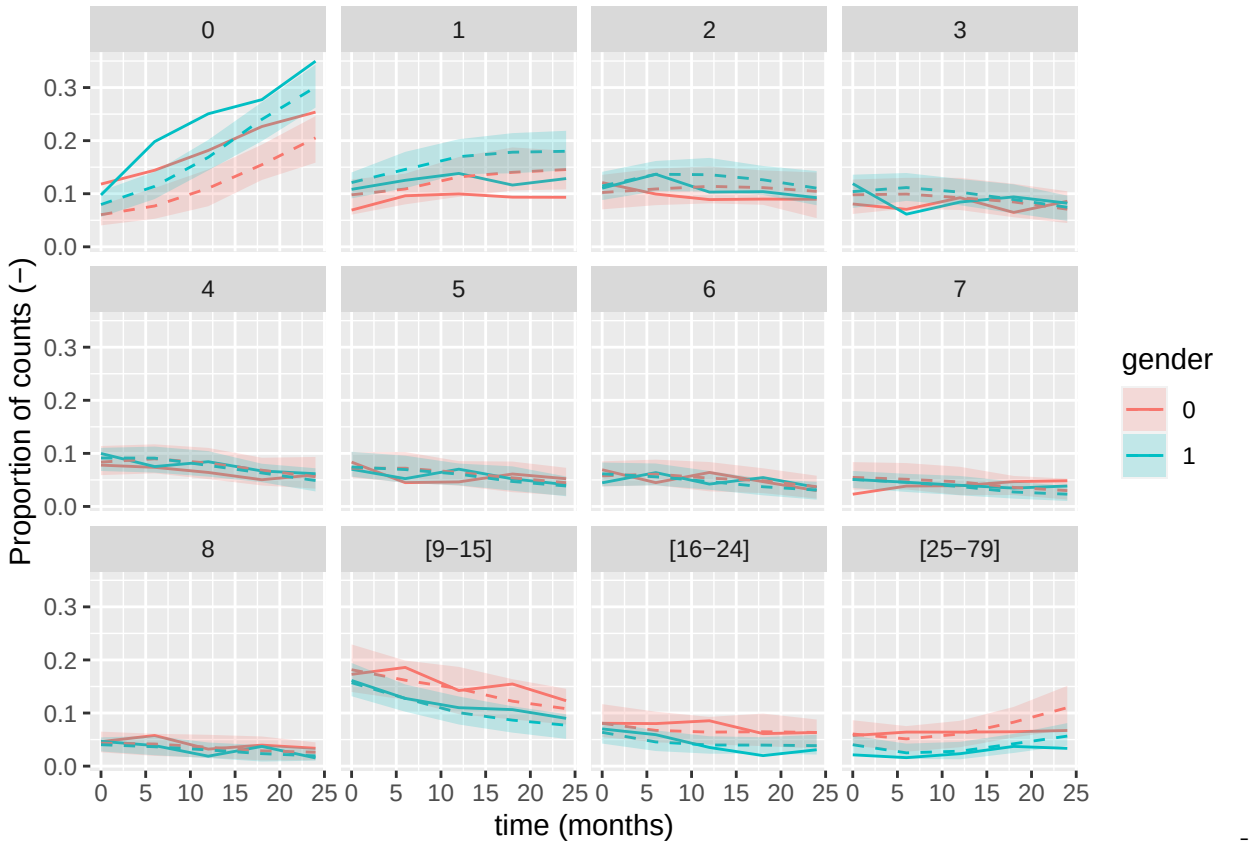
    - the population value of the Poisson parameter at baseline is 4.8
    - we see a decreasing trend with time, with the population value of λ after 2 years decreasing to 2.2
  - convergence plots show good convergence for all parameters

Comparing the parameter estimates from this fit to the estimates obtained by **glmer()** using a Laplace approximation in Table~2 of (Atkins et al. 2013), we find very good agreement with the SAEM algorithm.

Diagnostics 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.

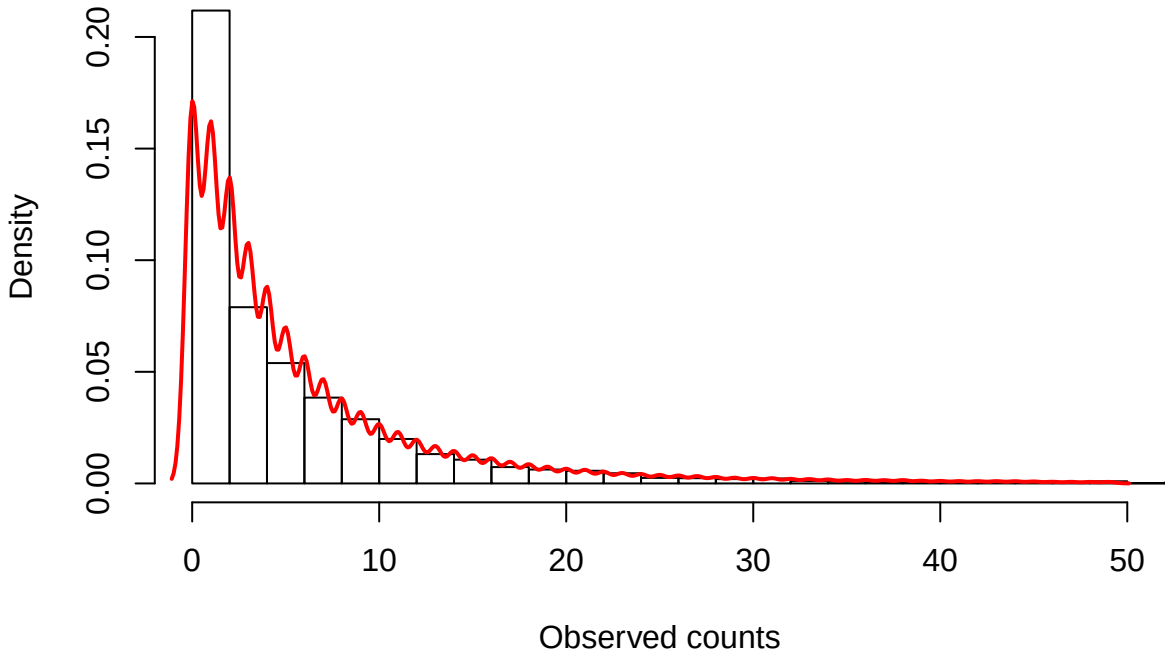
- Simulation function to simulate from a count model
  - the model function defines directly the log-pdf, so the user needs to define a function to simulate from the appropriate function
  - note the similarities between the model function (*count.poisson()*) and the simulation function (*countsimulate.poisson()*)
    - same setting of dependent variables (*time* and *rapi*) from *xidep* and parameters (*inter* and *slope*) from *psi*
      - note that we don't use *rapi* in *countsimulate.poisson()*
    - same definition of pevent ($=P(Y_{ij}=k)$, the probability of observing k counts)
    - in *count.poisson()* we then compute the probability of the observed outcome using the observed value of Y_{ij} contained in *rapi* for each observation
    - in *countsimulate.poisson()*, we use the individual value of $\lambda(Y_{ij})$ to simulate from a Poisson distribution using the *rpoisson()* function
- once the simulation function has been defined, we use the *simulateDiscreteSaemix()* function from the {saemix} package to simulate *nsim* values (here 100) with the population parameters estimated in *poisson.fit*
  - this adds a *simdata* element to the *poisson.fit*
  - we then use the *discreteVPC()* function to produce VPC corresponding to the fit

```
### Simulations
nsim<-100
yfit1<-simulateDiscreteSaemix(poisson.fit.cov2, nsim=nsim)
discreteVPC(yfit1, outcome="count", which.cov="gender", breaks=c(0:9, 16, 25,80))
```



we can also extract dataframe with the simulated data (*poisson.fit@sim.data@datasim*), adding a column *gender* to stratify the plots, in order to produce additional diagnostics

```
# Proportion of zeroes
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$rapi[yfit1@data@data$rapi==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])/lengt
```

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

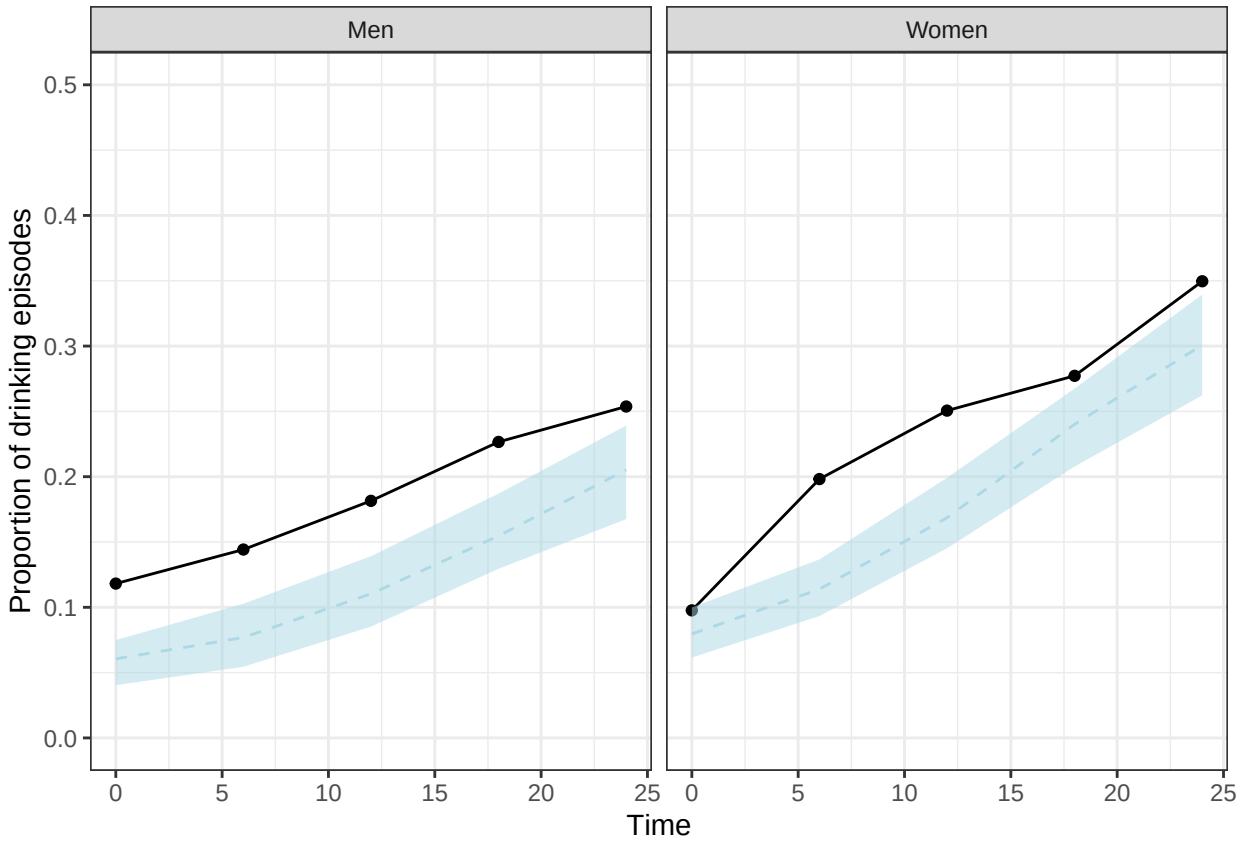
```
# Checking proportion of zeroes
yfit<-yfit1
simdat <-yfit@sim.data@datasim
simdat$time<-rep(yfit@data@data$time,nsim)
simdat$gender<-rep(yfit@data@data$gender,nsim)

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)
```



Dealing with overdispersion The Poisson model in the previous section 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. No variability is set on p_0 which represents a proportion at the level of the population and we use a logit-normal distribution for this parameter to ensure it remains between 0 and 1.

Other potential models include:

- the Generalised Poisson distribution
- the Negative Binomial model
- the Hurdle model (next section)

For the first two, we give expressions for the corresponding **saemix** models in the chunk code below, but not for the simulation functions which are more complicated (see package **RNGforGPD** for an implementation). The Hurdle model is given in the next section for comparison with the ZIP-Poisson model.

```
## Zero-inflated Poisson model
# Model
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)
}
# Simulation function
countsimulate.poissonzip<-function(psi, id, xidep) {
  time<-xidep[,1]
  y<-xidep[,2]
  ymax<-max(y)
  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
  y[y>ymax]<-ymax+1 # truncate to maximum observed value to avoid simulating aberrant values
  return(y)
}
## Generalized Poisson model with time effect
# Model
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)
}
# Simulation function - TBD, see RNGforGPD ?

## Negative binomial model with time effect
# Model
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)
}
# Simulation function - TBD ?
```

We then fit the ZIP model without and with covariates.

```
## ZIP base model
saemix.model.zip<-saemixModel(model=count.poissonzip,description="count model ZIP",modeltype="likelihood
                              simulate.function = countsimulate.poissonzip,
```

```
                              psi0=matrix(c(1.5, 0.01, 0.2),ncol=3,byrow=TRUE,dimnames=list(NULL, c("int
                              transform.par=c(0,0,3), covariance.model=diag(c(1,1,0)), omega.init=diag(

### ZIP Poisson with gender on both intercept
saemix.model.zip.cov1<-saemixModel(model=count.poissonzip,description="count model ZIP",modeltype="likel
                              simulate.function = countsimulate.poissonzip,
                                   psi0=matrix(c(1.5, 0.01, 0.2),ncol=3,byrow=TRUE,dimnames=list(NULL,
                                   transform.par=c(0,0,3), covariance.model=diag(c(1,1,0)), omega.init=
                                   covariate.model = matrix(c(1,0,0),ncol=3, byrow=TRUE), verbose=FALSE)
### ZIP Poisson with gender on both intercept and slope
saemix.model.zip.cov2<-saemixModel(model=count.poissonzip,description="count model ZIP",modeltype="likel
                              simulate.function = countsimulate.poissonzip,
                                   psi0=matrix(c(1.5, 0.01, 0.2),ncol=3,byrow=TRUE,dimnames=list(NULL,
                                   transform.par=c(0,0,3), covariance.model=diag(c(1,1,0)), omega.init=
                                   covariate.model = matrix(c(1,1,0),ncol=3, byrow=TRUE), verbose=FALSE)

zippoisson.fit<-saemix(saemix.model.zip,saemix.data,saemix.options)
zippoisson.fit.cov1<-saemix(saemix.model.zip.cov1,saemix.data,saemix.options)
zippoisson.fit.cov2<-saemix(saemix.model.zip.cov2,saemix.data,saemix.options)
print(zippoisson.fit.cov2@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
```

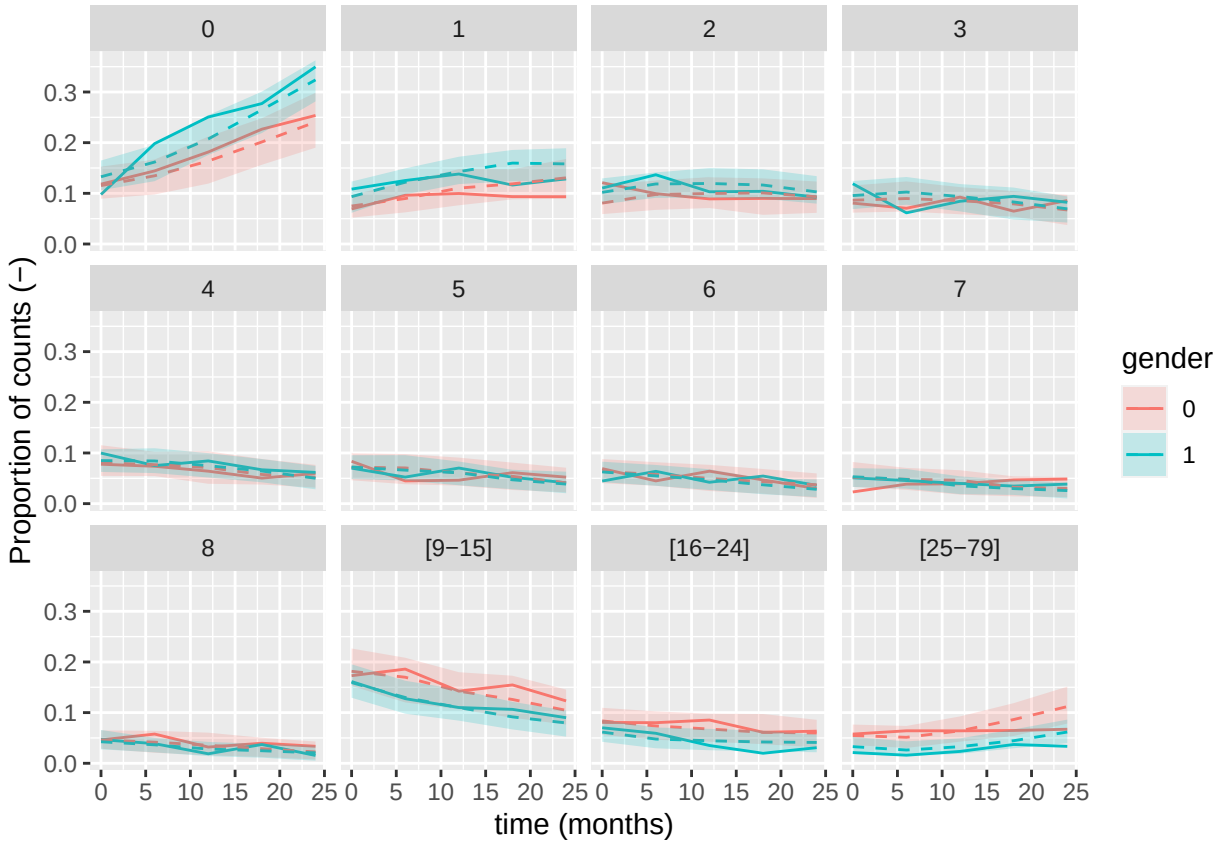
```
# saemix.model.gen<-saemixModel(model=count.genpoisson,description="count model Generalised Poisson",mo
```

The results from the covariate model are very close to the previous ones for the common parameters (intercept, slope and gender effects). The proportion of overinflated 0.075.

Diagnostics We can look at VPC plots for the different values of the score. Here, we plot separate VPC for scores 0 to 9 then regroup the higher scores into 3 categories as the number of subjects with more than 10 drinking episodes drops quickly. The code below can be tweaked to adjust to different score categories if needed (changing the *breaks* argument). The ZIP model is able to predict the probability of observing different values of the score in both men and women over time.

We compare the previous fit from the Poisson model with the ZIP model using the same approach as before, and find a much better agreement between the predictions and the fit in both genders.

```
### Simulations
ysim.zip2<-simulateDiscreteSaemix(zippoisson.fit.cov2,  100)
discreteVPC(ysim.zip2, outcome="count", breaks=c(0:9,16,25,80), which.cov="gender")
```



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

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

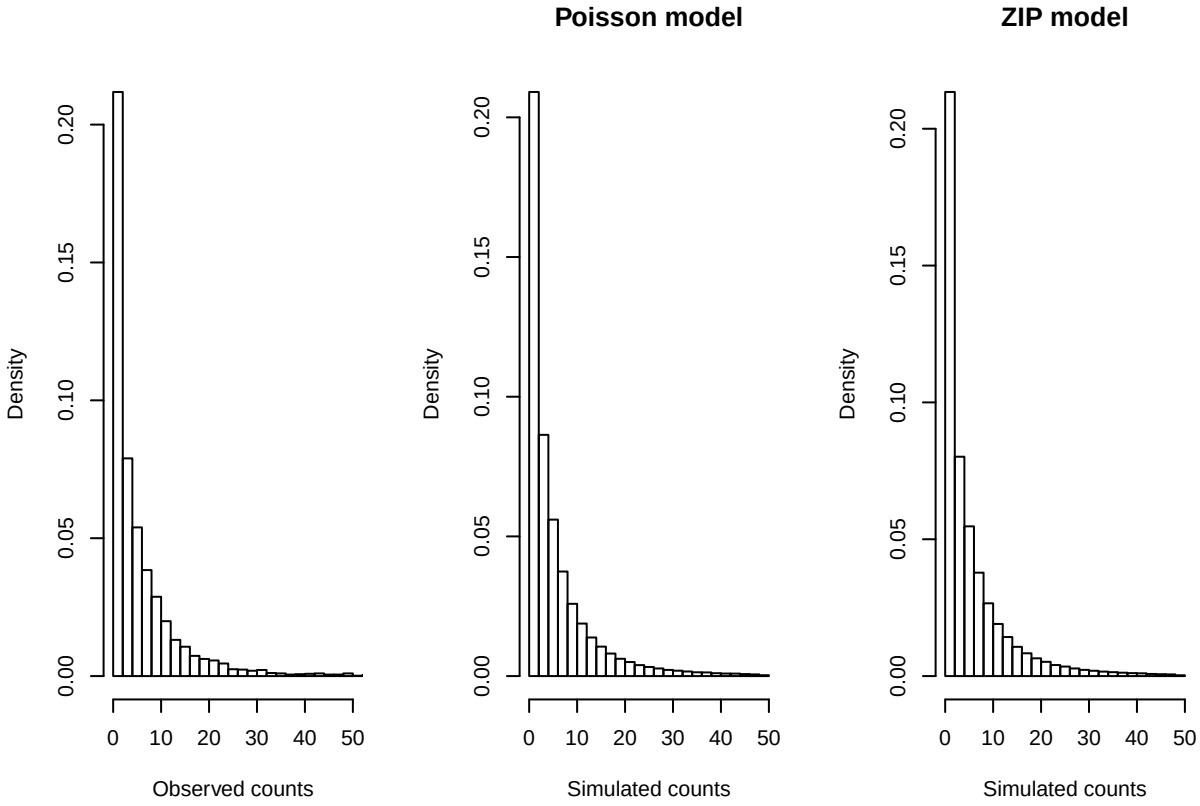
```
cat("      Poisson model, p=",length(yfit1@sim.data@datasim$ysim[yfit1@sim.data@datasim$ysim==0])/lengt
```

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

```
cat("  ZI-Poisson model, p=",length(ysim.zip2@sim.data@datasim$ysim[ysim.zip2@sim.data@datasim$ysim==0]
```

```
##   ZI-Poisson model, p= 0.1957329
```

```
par(mfrow=c(1,3))
hist(yfit1@data@data$rapi, xlim=c(0,50), freq=F, breaks=30, xlab="Observed counts", main="")
hist(yfit1@sim.data@datasim$ysim[yfit1@sim.data@datasim$ysim<50], xlim=c(0,50), freq=F, breaks=20, xlab
hist(ysim.zip2@sim.data@datasim$ysim[ysim.zip2@sim.data@datasim$ysim<50], xlim=c(0,50), freq=F, breaks=
```




```
# Checking proportion of zeroes
yfit<-ysim.zip2
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) %>%
```

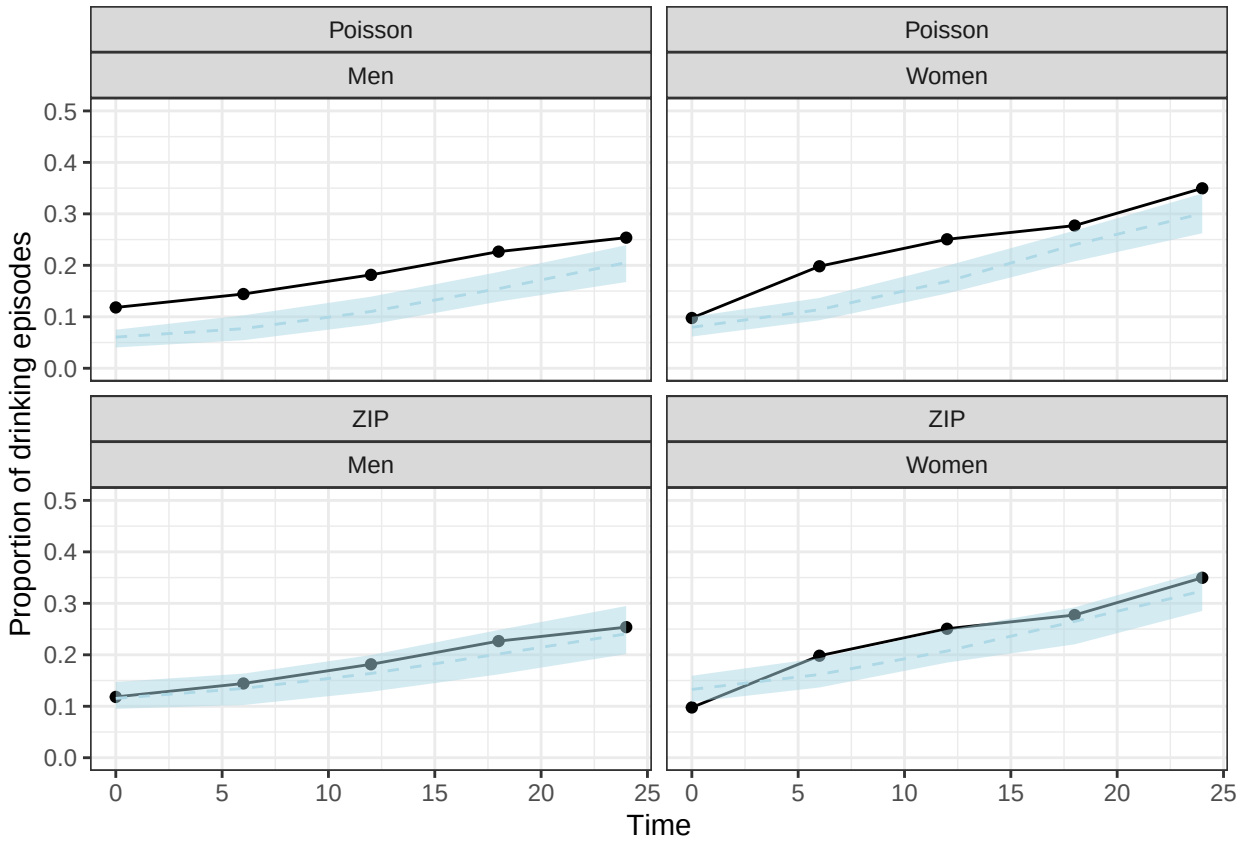
```
    summarise(nev = sum(ysim==0), n=n()) %>%
    mutate(freq = nev/n)
  )
  ytab<-rbind(ytab,xtab1[,c("time","gender","freq")])
}
gtab <- ytab %>%
    group_by(time, gender) %>%
  summarise(lower=quantile(freq, c(0.05)), median=quantile(freq, c(0.5)), upper=quantile(freq, c(0.95)))
  mutate(gender=ifelse(gender==0,"Men","Women"))
```

```
## `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)

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(model~gender) +
  xlab("Time") + ylab("Proportion of drinking episodes")

print(plot2)
```



Hurdle model Another way to handle the excess of 0's is the so-called hurdle model. In this model, we first fit a binary model to the dichotomised data (no event versus an event), then a Poisson model to the subjects who experienced at least one event. Compared to the ZIP model we fitted previously, this model lets the proportion of subjects without alcohol-related events vary with time through the binary logistic regression part (see more details on binary/categorical models in the corresponding notebook).

```
## 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("")), verbose=FALSE)

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("")), verbose=FALSE)

# 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)
}
# Associated simulation function
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)), verb

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

hurdlefit0<-saemix(saemix.hurdle0,saemix.data0,saemix.options)
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="likeli
                                 simulate.function = countsimulate.poisson,
                                 psi0=matrix(c(log(5),0.01),ncol=2,byrow=TRUE,dimnames=list(NULL, c("int
                                 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), verbose=FALSE)
saemix.options<-list(seed=632545,save=FALSE,save.graphs=FALSE, displayProgress=FALSE, print=FALSE)

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

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

```
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@s
```

```
## Simulated proportion of 0's overall: 0.2069994
```

```
ysim.hurdle1 <- simulateDiscreteSaemix(hurdlefit1, nsim=nsim)

# 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)
```

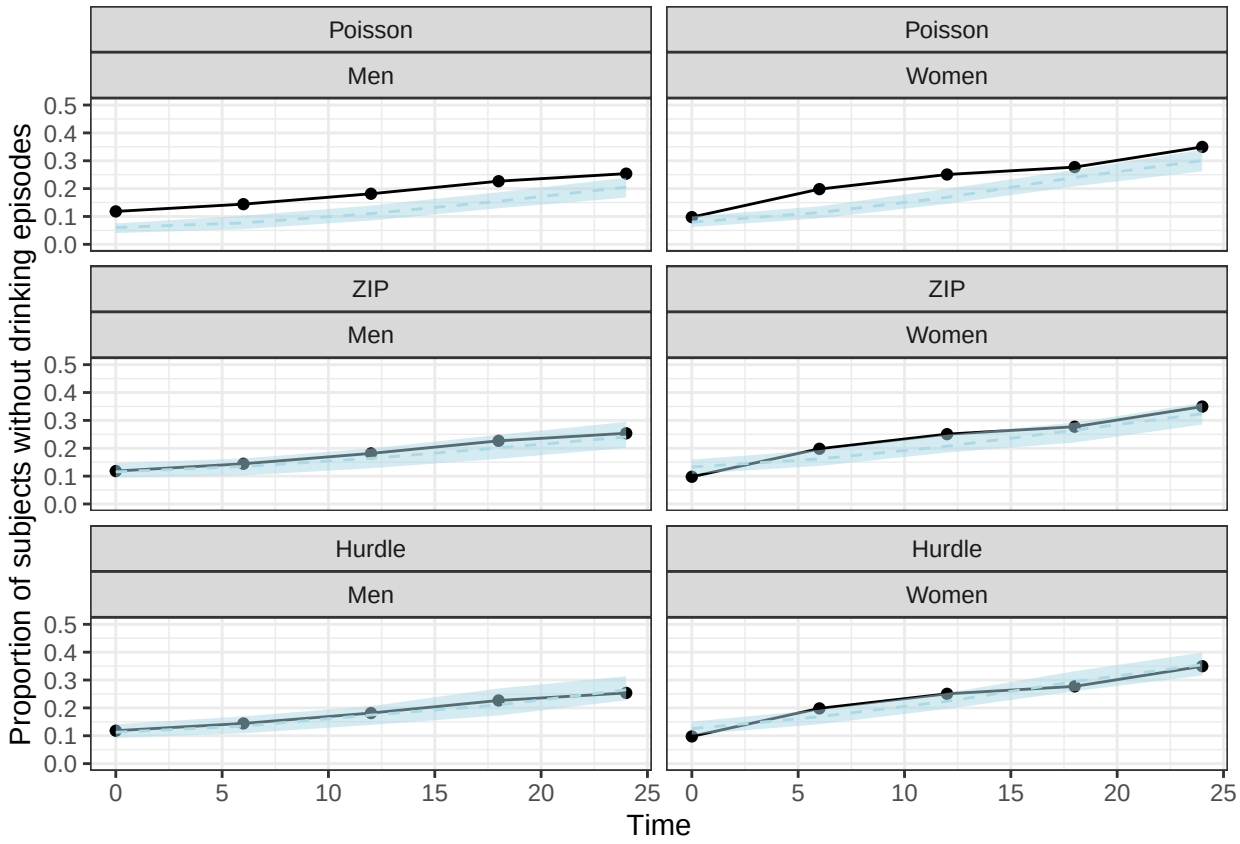
```
## `summarise()` has grouped output by 'time'. You can override using the `.groups` argument.
```

```
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","om
                   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
```

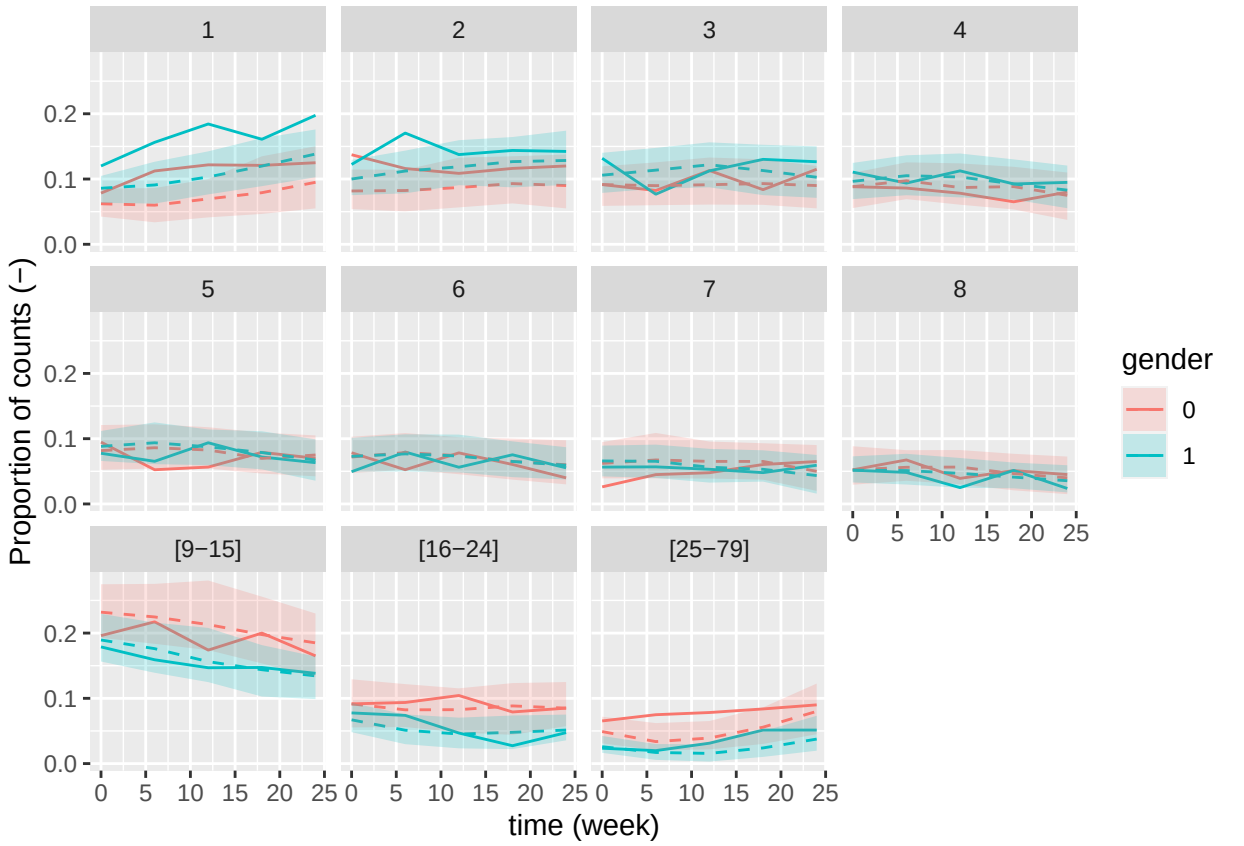
Comparing the proportion of 0's for the different models We have already seen a clear model misfit for Poisson, on the other hand the other two models predict the proportion of subjects without drinking episodes much more accurately, with no clear difference between them.

```
plot.prop0 <- 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(plot.prop0)
```



We can also produce VPC for the counts as previously for the ZIP model.

```
nsim<-100
ysim.hurdle1 <- simulateDiscreteSaemix(hurdlefit1, nsim=nsim)
discreteVPC(ysim.hurdle1, outcome="count", breaks=c(0:9, 16, 25, 80), which.cov="gender")
```



Standard errors of estimation The computation of the FIM in **saemix** uses the so-called FOCE method, an approximation where the model function f is linearised around the conditional expectation of the individual parameters. This approximation is particularly bad for discrete data models, which is why currently **saemix** doesn't provide estimation errors for count data models. For non-Gaussian models, the exact FIM should be computed, and two approaches have been proposed using either numerical integration by a combination of MC and adaptive Gaussian quadrature (MC/AGQ, Ueckert et al 2017) or stochastic integration by MCMC (Rivière et al. 2017). Both these approaches are computationally intensive. They require to define the elementary FIM associated to the observed design (here, the number of times and the gender which enters the model), and sum over all the subjects sharing the same design. Here, as in the *toenail* example, the subjects in the real dataset *rapi.saemix* were not all seen at the same times. This makes the computation of an exact FIM somewhat cumbersome, especially when stratifying on gender, as we need to compute the FIM in each combination of gender and observation times and sum the elementary FIM. We leave that to th

Alternatively, different bootstrap approaches can be used in non-linear mixed effect models and have been implemented for **saemix** in Comets et al. 2021, with code available on the github for the package (https://github.com/saemixdevelopment/saemixextension).

```
# Time-patterns
time.pattern <- tapply(rapi.saemix$time, rapi.saemix$id, function(x) paste(x,collapse="-"))
table(time.pattern)
```

```
## time.pattern
##             0          0-12       0-12-18    0-12-18-24       0-12-24          0-18
##            25             4             4            18             3             4
##       0-18-24          0-24          0-6       0-6-12     0-6-12-18 0-6-12-18-24
##             6             3            27            36            54           561
##     0-6-12-24       0-6-18    0-6-18-24       0-6-24
##            28             7            28            10
```

```r
# Time-patterns by gender
print(table(time.pattern, rapi.saemix$gender[!duplicated(rapi.saemix$id)]))
```

```
##
## time.pattern   Men Women
##   0             15    10
##   0-12           2     2
##   0-12-18        2     2
##   0-12-18-24     8    10
##   0-12-24        2     1
##   0-18           2     2
##   0-18-24        4     2
##   0-24           0     3
##   0-6           15    12
##   0-6-12        17    19
##   0-6-12-18     22    32
##   0-6-12-18-24 213   348
##   0-6-12-24     15    13
##   0-6-18         4     3
##   0-6-18-24     23     5
##   0-6-24         3     7
```

Case bootstrap: The first bootstrap approach we can use is case bootstrap, where we resample at the level of the individual. We plot the bootstrap distribution for the 4 parameters (intercept, slope, treatment effect on slope, and variability of intercept). The red vertical line represents the estimate obtained on the original data while the blue line shows the mean of the bootstrap distribution.

Conditional bootstrap: We can also use conditional bootstrap, a non-parametric residual bootstrap which bootstraps samples from the conditional distributions and preserves the exact structure of the original dataset.

Bootstrap CI Here we load the results from the two bootstrap files prepared beforehand by running the *saemix.bootstrap* code with 500 simulations for the ZIP model with covariates. We compute the bootstrap quantiles for the 95% CI, as well as the SD of the bootstrap distribution, corresponding to a normal approximation of the SE. Both bootstrap methods give similar estimates for the CI's of the different parameters.

```r
if(runBootstrap) {
  case.count <- saemix.bootstrap(zippoisson.fit.cov2, method="case", nboot=nboot)
  cond.count <- saemix.bootstrap(zippoisson.fit.cov2, method="conditional", nboot=nboot)
} else {
 case.count <- read.table(file.path(saemixDir,"bootstrap","results","rapi_caseBootstrap.res"), header=T
 cond.count <- read.table(file.path(saemixDir,"bootstrap","results","rapi_condBootstrap.res"), header=T
 nboot<-dim(case.count)[1]
}
case.count <- case.count[!is.na(case.count[,2]),]
cond.count <- cond.count[!is.na(cond.count[,2]),]

par.estim<-c(zippoisson.fit.cov2@results@fixed.effects,diag(zippoisson.fit.cov2@results@omega)[zippoiss
df2<-data.frame(parameter=colnames(case.count)[-c(1)], saemix=par.estim)
for(i in 1:2) {
  if(i==1) {
    resboot1<-case.count
    namboot<-"case"
  } else {
    resboot1<-cond.count
```

```
    namboot <-"cNP"
  }
  mean.bootDist<-apply(resboot1, 2, mean)[-c(1)]
  sd.bootDist<-apply(resboot1, 2, sd)[-c(1)]
  quant.bootDist<-apply(resboot1[-c(1)], 2, quantile, c(0.025, 0.975))
  l1<-paste0(format(mean.bootDist, digits=2)," (",format(sd.bootDist,digits=2, trim=T),")")
  l2<-paste0("[",format(quant.bootDist[1,], digits=2),", ",format(quant.bootDist[2,],digits=2, trim=T),
  df2<-cbind(df2, l1, l2)
  i1<-3+2*(i-1)
  colnames(df2)[i1:(i1+1)]<-paste0(namboot,".",c("estimate","CI"))
}
print(df2)
```

```
##                parameter       saemix     case.estimate            case.CI
## 1              intercept  1.772785031  1.7677 (0.05781)  [ 1.6532, 1.8840]
## 2 beta_gender.intercept. -0.196792601 -0.1991 (0.07404) [-0.3434, -0.0565]
## 3                  slope -0.020435421 -0.0196 (0.00384) [-0.0276, -0.0121]
## 4     beta_gender.slope. -0.015721267 -0.0155 (0.00473) [-0.0248, -0.0064]
## 5                     p0  0.075407726  0.0763 (0.00765)  [ 0.0627, 0.0916]
## 6       omega2.intercept  0.782636954  0.7908 (0.05303)  [ 0.6890, 0.8964]
## 7           omega2.slope  0.003255963  0.0032 (0.00037)  [ 0.0024, 0.0039]
##        cNP.estimate             cNP.CI
## 1  1.7825 (0.05076)  [ 1.6793, 1.8798]
## 2 -0.1960 (0.06865) [-0.3315, -0.0722]
## 3 -0.0202 (0.00393) [-0.0275, -0.0122]
## 4 -0.0148 (0.00496) [-0.0255, -0.0055]
## 5  0.0770 (0.00553)  [ 0.0663, 0.0872]
## 6  0.7552 (0.04654)  [ 0.6702, 0.8484]
## 7  0.0028 (0.00024)  [ 0.0023, 0.0032]
```

References

Atkins D, Baldwin S, Zheng C, Gallop R, Neighbors C (2013). A tutorial on count regression and zero-altered count models for longitudinal substance use data. *Psychology of Addictive Behaviors* 27:166–77.

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

Neighbors CJ, Lewis M, Atkins D, Jensen M, Walter T, Fossos N, Lee C, Larimer M (2010a). Efficacy of web-based personalized normative feedback: A two-year randomized controlled trial. *Journal of Consulting and Clinical Psychology* 78:898–911.

Neighbors CJ, Barnett NP, Rohsenow DJ, Colby SM, and Monti PM (2010b). Cost-effectiveness of a motivational lntervention for alcohol-involved youth in a hospital emergency department. *Journal of Studies on Alcohol and Drugs* 71:384–94.

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

White HR, Labouvie EW (1989). Towards the assessment of adolescent problem drinking. *Journal of Studies on Alcohol* 50:30–7.