Saemix 3 - binary and categorical 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_categoricalModel.R* in the same folder.

Binary response model

Data description The *toenail.saemix* dataset in the **saemix** package contains binary data from a randomised clinical trial comparing two treatments for fungal toenail infection. The original data is available in **R** as the *toenail* dataset in the package **prLogistic** and has been reformatted for **saemix**.

The data was collected in a multi-center randomised comparison of two oral treatments (A and B) for toenail infection. 294 patients are measured at seven visits, i.e. at baseline (week 0), and at weeks 4, 8, 12, 24, 36, and 48 thereafter, comprising a total of 1908 measurements. The primary end point was the presence of toenail infection and the outcome of interest is the binary variable "onycholysis" which indicates the degree of separation of the nail plate from the nail-bed (0=none or mild versus 1=moderate or severe).

To create the data object using saemixData, we need to specify the response column both as a response (*name.response="y"*) and as a predictor (here, time is the first predictor and we add the response in the argument *name.predictors*).

```
data(toenail.saemix)
```

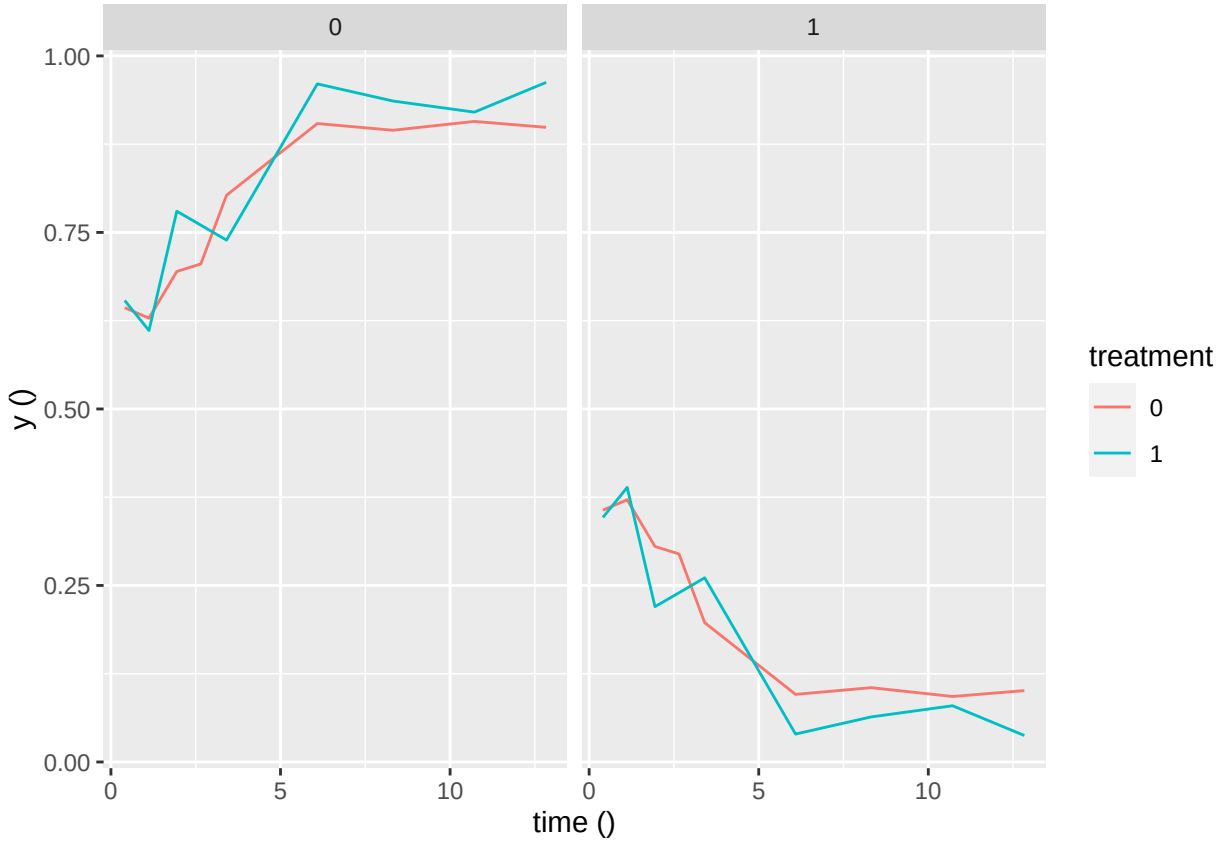
```
saemix.data<-saemixData(name.data=toenail.saemix,name.group=c("id"),name.predictors=c("time","y"), name
                        name.covariates=c("treatment"), verbose=FALSE)
```

Exploring data The usual plot of the data object is not very informative as it alternates between 0 and 1's. Instead we plot the evolution of the frequency of infection over time in the population, stratifying by treatment. We use the *plotDiscreteData()* function from the package, setting the *outcome* argument to *'binary'*.

```
# Distribution of times
if(FALSE) hist(toenail.saemix$time, breaks=c(-1,0.25,1.25,2.25, 3.25, 7,10,15,20), freq=T)
table(cut(toenail.saemix$time, breaks=c(-1,0.25,1.25,2.25, 3.25, 7,10,15,20)))
```

```
##
##   (-1,0.25] (0.25,1.25] (1.25,2.25] (2.25,3.25]    (3.25,7]      (7,10]
##         294         278         272         240         291         245
##     (10,15]     (15,20]
##         282           6
```

```
# Proportion of 0's and 1's across time
plotDiscreteData(saemix.data, outcome='binary', which.cov="treatment")
```



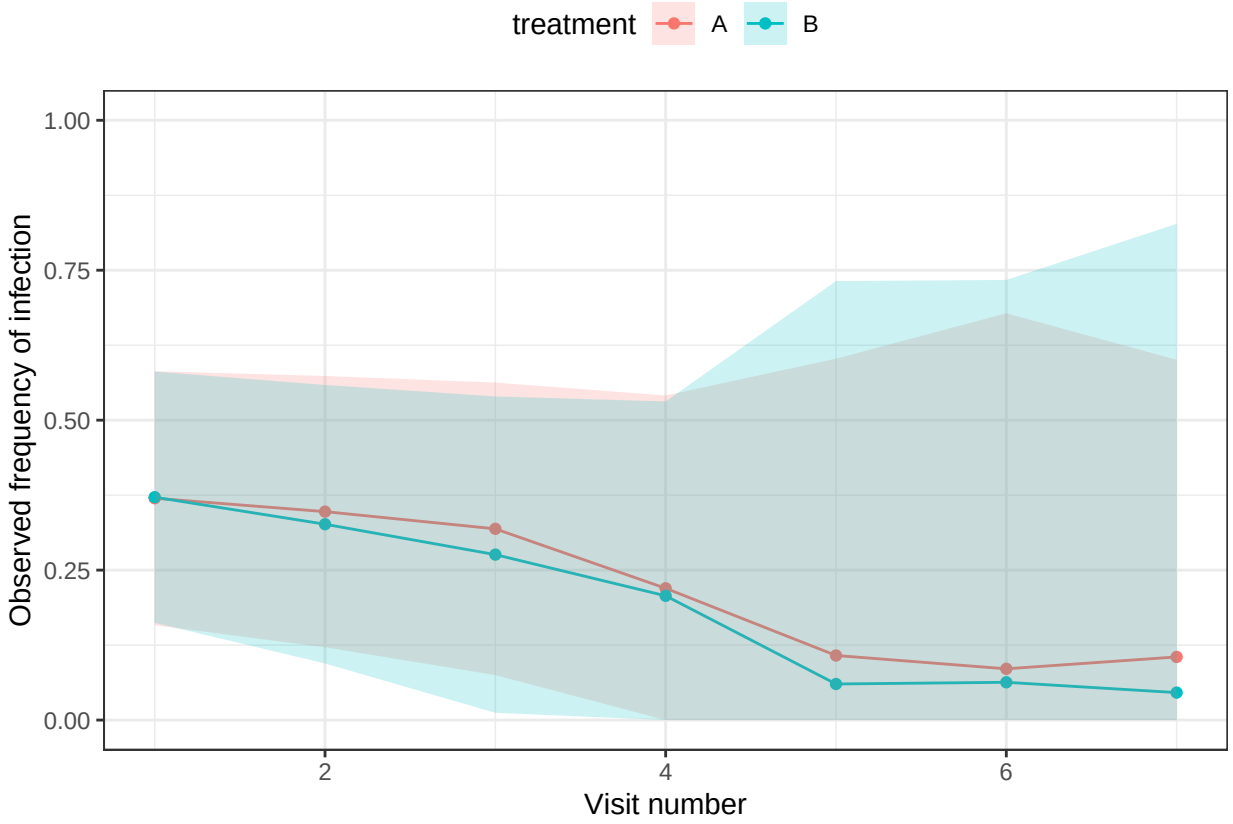
We can also present the results as a barplot of the proportion of events, using **ggplot2** and **tidyverse**.

```
# Barplots across time
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")

print(plot1)
```




```
if(saveFigs) {
  namfig<-"toenail_infectionFreq.eps"
  cairo_ps(file = file.path(figDir, namfig), onefile = TRUE, fallback_resolution = 600, height=8.27, wi
  plot(plot1)
  dev.off()
}
```

Statistical model The model fit here is a logistic random effect model developed by (Hedeker et al. 1994). This model includes a random intercept (θ_1 and ω_1), a time effect (θ_2) and treatment (A or B) (β) as a covariate affecting the slope θ_2. We considered the interaction term between time and treatment but no treatment effect alone as it would impact the intercept which shouldn't be different between arms due to the randomisation process. The time course was assumed to be similar across subjects as no significant interindividual variability was found when testing different models.

- Model for repeated binary data
  - the probability that y_{ij} outcome observed in subject i at visit j (time t_{ij} in months) is 1 is modelled as a logistic model
  - linear model on the logit scale ($\text{logit}(p) = \ln\left(\frac{p}{1-p}\right)$)

$$\text{logit}(P(y_{ij} = 1)) = \theta_{1,i} + \theta_{2,i} t_{ij}$$

- Statistical model
  - $\theta_{1,i}$ assumed to follow a normal distribution $N(\mu_{\theta_1}, \omega_{\theta_1})$
  - $\theta_{2,i}$ assumed to depend on treatment as in $\theta_{2,i} = \mu_\beta + 1_{trt=B}$

In the following chunk of code, we define the model and the simulate function, then create the saemixModel object.

```
# 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)
}

# simulation function (used for diagnostics)
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, n
                          psi0=matrix(c(-0.5,-.15,0,0),ncol=2,byrow=TRUE,dimnames=list(NULL,c("theta1","
                          transform.par=c(0,0), covariate.model=c(0,1),covariance.model=matrix(c(1,0,0,0
```

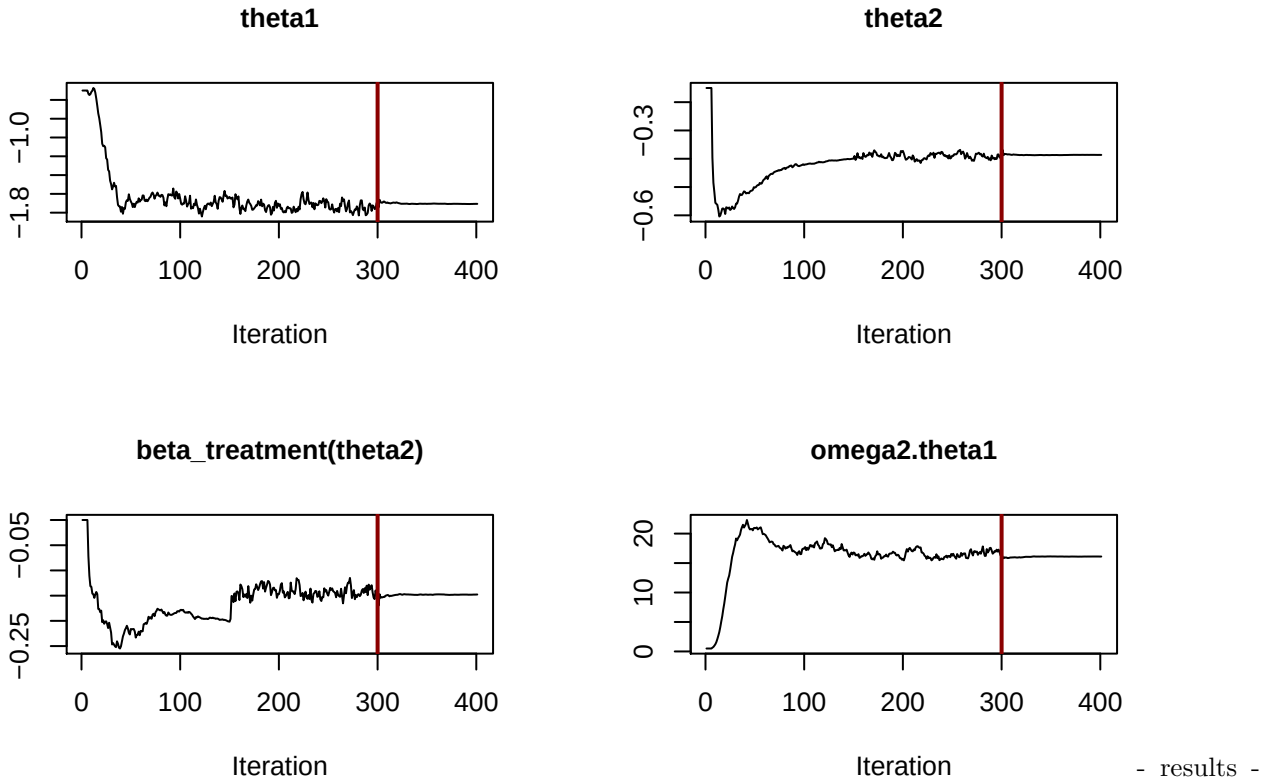
- Fit with saemix and store the results to the object *binary.fit*
  - setting options
  - 10 chains
  - don't display intermediate graphs or save graphs
  - don't compute the FIM (approximation not suited to discrete data models)

```
# saemix fit
saemix.options<-list(seed=1234567,save=FALSE,save.graphs=FALSE, displayProgress=FALSE, nb.chains=10, fi
binary.fit<-saemix(saemix.model,saemix.data,saemix.options)
summary(binary.fit)
```

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

```
## -----------------  Fixed effects  ------------------
## ----------------------------------------------------
##                 Parameter Estimate
## 1                    theta1    -1.71
## 2                    theta2    -0.39
## 3 beta_treatment(theta2)    -0.15
## ----------------------------------------------------
## -----------  Variance of random effects  -----------
## ----------------------------------------------------
##            Parameter Estimate
## theta1 omega2.theta1    16.11
## ----------------------------------------------------
## ------  Correlation matrix of random effects  ------
## ----------------------------------------------------
##               omega2.theta1
## omega2.theta1 1.00
## ----------------------------------------------------
## --------------  Statistical criteria  -------------
## ----------------------------------------------------
##
## Likelihood computed by importance sampling
##       -2LL= 1250.678
##       AIC = 1260.678
##       BIC = 1279.096
## ----------------------------------------------------
```

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



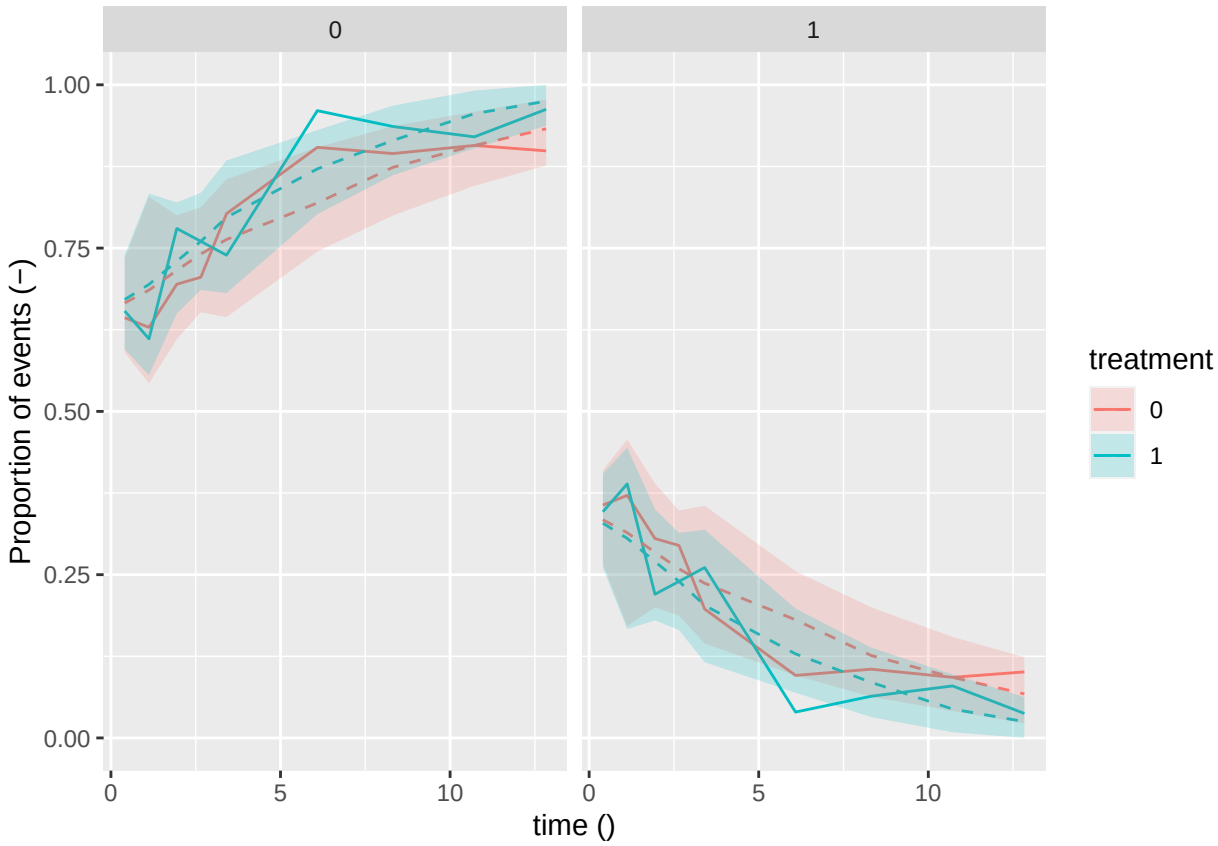
- results - numerical output - note that the estimated value of μ_{θ_1} is -1.71, corresponding to an estimated probability of event of 0.154 - this is lower than the observed probability of infection at time 0 (around 0.37)

because the logistic model is highly non-linear and $E(f(\theta))$ is very different from $f(E(\theta))$ - convergence plots show good convergence for all parameters

Diagnostics

- Simulation function to simulate from a binary model
  - the model function defines directly the log-pdf, so the user needs to define a function to simulate from the appropriate binomial function
  - note the similarities between the model function (*binary.model()*) and the simulation function (*simulBinary()*)
    - same setting of dependent variables (*tim* and *y*) from *xidep* and parameters (*inter* and *slope*) from *psi*
      - note that we don't use *y* in *simulBinary()*
    - same definition of pevent ($=P(Y_{ij}=1)$, the probability of observing an event)
    - in *binary.model()* we then compute the probability of the observed outcome using the observed value of Y_{ij} contained in *y* for each observation
    - in *simulBinary()*, we use the individual $P(Y_{ij}=1)$ predictions to simulate from a Bernouilli distribution using the *rbinom()* function
- once the simulation function has been defined and associated with the model component of the object, we use the *simulateDiscreteSaemix()* function from the {saemix} package to simulate *nsim* values (here 100) with the population parameters estimated in *binary.fit*
  - this adds a *simdata* element to the *binary.fit*
  - the simulations are used to produce VPC via the *discreteVPC()* function, again specifying the outcome to be binary. We stratify the plot on the treatment covariate

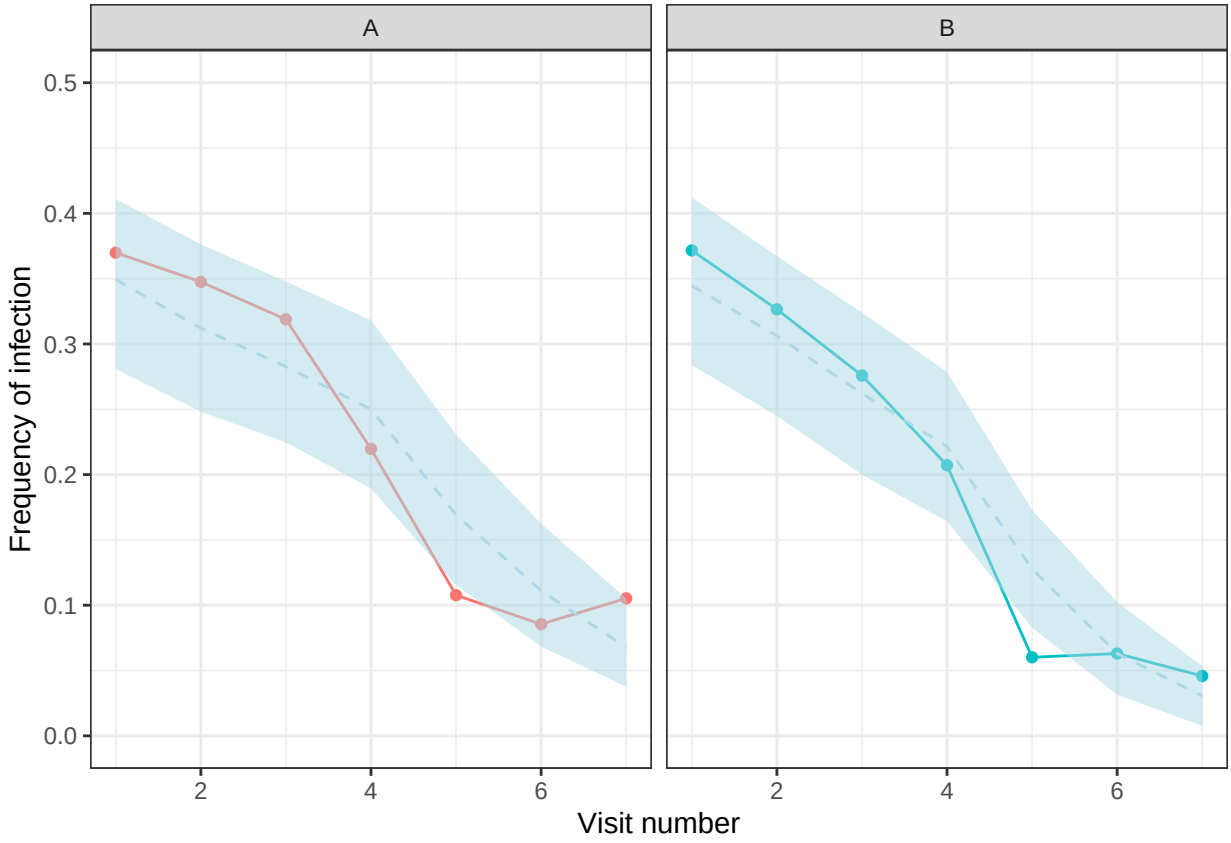
```
#  $1_{Y_{ij}=0} \times (1-P(Y_{ij}=1)) + 1_{Y_{ij}=1} \times P(Y_{ij}=1) $
# simulate from model (nsim=100)
nsim<-1000
binary.fit <- simulateDiscreteSaemix(binary.fit, nsim=nsim)

discreteVPC(binary.fit, outcome="binary", which.cov="treatment")
```



We can also extract dataframe with the simulated data (*binary.fit@sim.data@datasim*) and produce diagnostics in R. Below, using **tidyverse**, we add columns *visit* and *treatment* to plot the frequency of infection over time for each treatment.

```
simdat <-binary.fit@sim.data@datasim
simdat$visit<-rep(toenail.saemix$visit,nsim)
simdat$treatment<-rep(toenail.saemix$treatment,nsim)
# VPC-type diagnostic
ytab<-NULL
for(irep in 1:nsim) {
  xtab<-simdat[simdat$irep==irep,]
  suppressMessages(
  xtab1 <- xtab %>%
    group_by(visit, treatment) %>%
    summarise(nev = sum(ysim), n=n()) %>%
    mutate(freq = nev/n)
  )
  ytab<-rbind(ytab,xtab1[,c("visit","freq","treatment")])
}
gtab <- ytab %>%
  group_by(visit, treatment) %>%
  summarise(lower=quantile(freq, c(0.05)), median=quantile(freq, c(0.5)), upper=quantile(freq, c(0.95)))
  mutate(treatment=ifelse(treatment==1,"B","A"))
gtab$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(saveFigs) {
  namfig<-"toenail_vpcByTreatment.eps"
  cairo_ps(file = file.path(figDir, namfig), onefile = TRUE, fallback_resolution = 600, height=8.27, wi
  plot(plot2)
  dev.off()
}
```

- npde for categorical data (submitted)
  - **TODO** using code from Marc

```
# npd
```

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 categorical/binary data models. In this document we show how to obtain SE through the computation of the exact FIM using numerical integration, as well as a bootstrap approach. Because all subjects have different times, for the binary data we will use bootstrap approaches as computing the exact FIM is time-consuming and we would need to compute it for each subject separately before summing the individual FIMs.

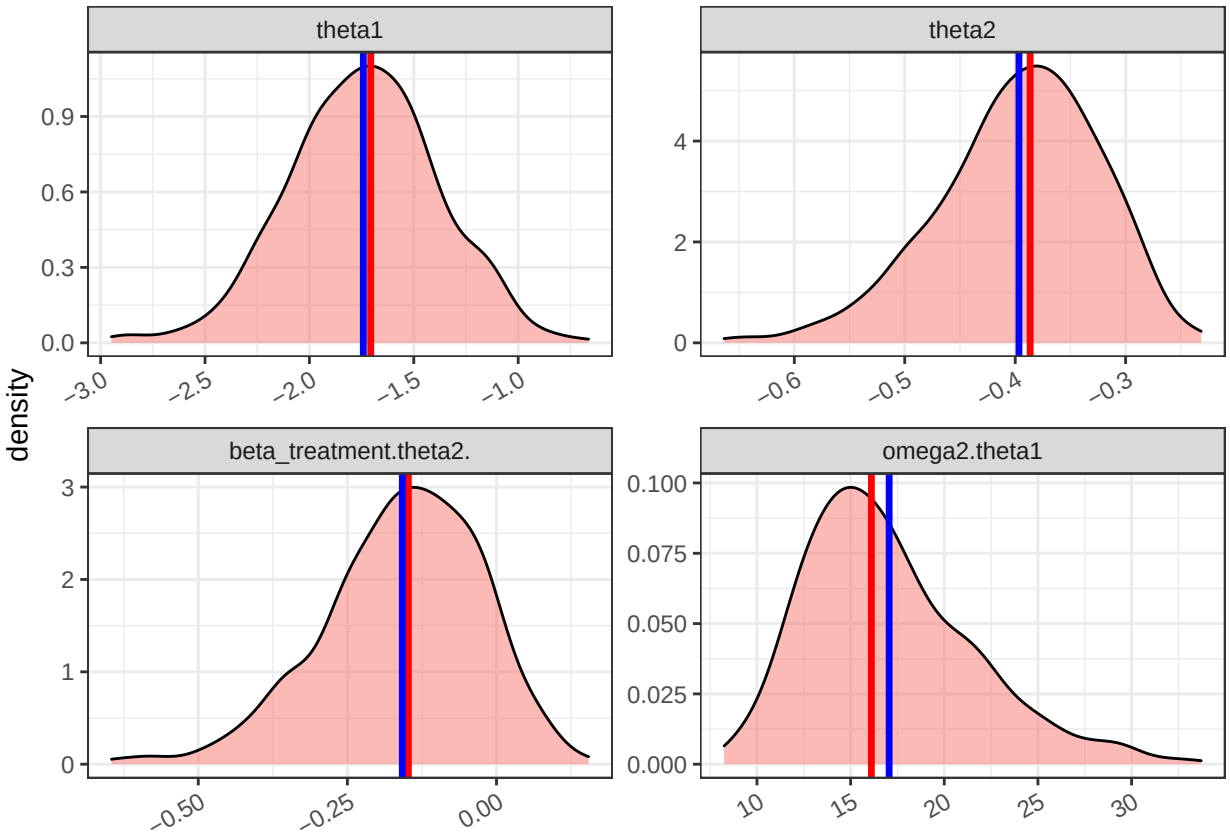
Different bootstrap approaches can be used in non-linear mixed effect models and have been implemented for **saemix** in Comets et al. 2021, with code available on the github.

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.

```
if(!runBootstrap)  {
  case.bin <- read.table(file.path(saemixDir,"bootstrap","results","toenail_caseBootstrap.res"), header=
  nboot<-dim(case.bin)[1]
}  else case.bin <- saemix.bootstrap(binary.fit, method="case", nboot=nboot)
head(case.bin)
```

```
##   Replicate    theta1     theta2 beta_treatment.theta2. omega2.theta1
## 1         1 -1.851662 -0.2905759            -0.21093899      13.21270
## 2         2 -1.582608 -0.3956728            -0.14496853      21.50126
## 3         3 -1.290633 -0.4133164            -0.43242325      23.04858
## 4         4 -1.851579 -0.4228992            -0.17280887      19.78489
## 5         5 -1.453970 -0.4218880            -0.01963166      11.77697
## 6         6 -1.728966 -0.3965398            -0.18952473      13.28420
```

```
# Bootstrap distributions
#if(nboot<200) cat("The number of bootstrap samples is too low to provide good estimates of the confide
  resboot1<-case.bin
  ypd2<-NULL
  for(icol in 1:4) {
    ypd2<-rbind(ypd2,data.frame(rep=resboot1[,1],Param=colnames(resboot1)[(icol+1)],value=resboot1[,(ic
  }

  ypd2$Param<-factor(ypd2$Param, levels = unique(ypd2$Param))
  ypd2.fix<-ypd2[ypd2$Param %in% unique(ypd2$Param)[1:3],]
  ypd2.iiv<-ypd2[ypd2$Param %in% unique(ypd2$Param)[4],]
  ypd <- ypd2

  par.estim<-c(binary.fit@results@fixed.effects,diag(binary.fit@results@omega)[binary.fit@results@indx.
  mean.bootDist<-apply(resboot1, 2, mean)[-c(1)]
  df<-data.frame(Param=unique(ypd2$Param), mean.boot=mean.bootDist, est.saemix=par.estim, Bootstrap="Cas

  plot.density2<-ggplot(data=ypd2) + geom_density(aes(value,fill="red4"), alpha=0.5) +
    geom_vline(data=df,aes(xintercept=est.saemix),colour="red",size=1.2) +
    geom_vline(data=df,aes(xintercept=mean.boot),colour="blue",size=1.2) +
    theme_bw() + theme(axis.title.x = element_blank(),axis.text.x = element_text(size=9, angle=30, hjus
    facet_wrap(~Param, ncol=2, scales = 'free')

  print(plot.density2)
```




```
#}
```

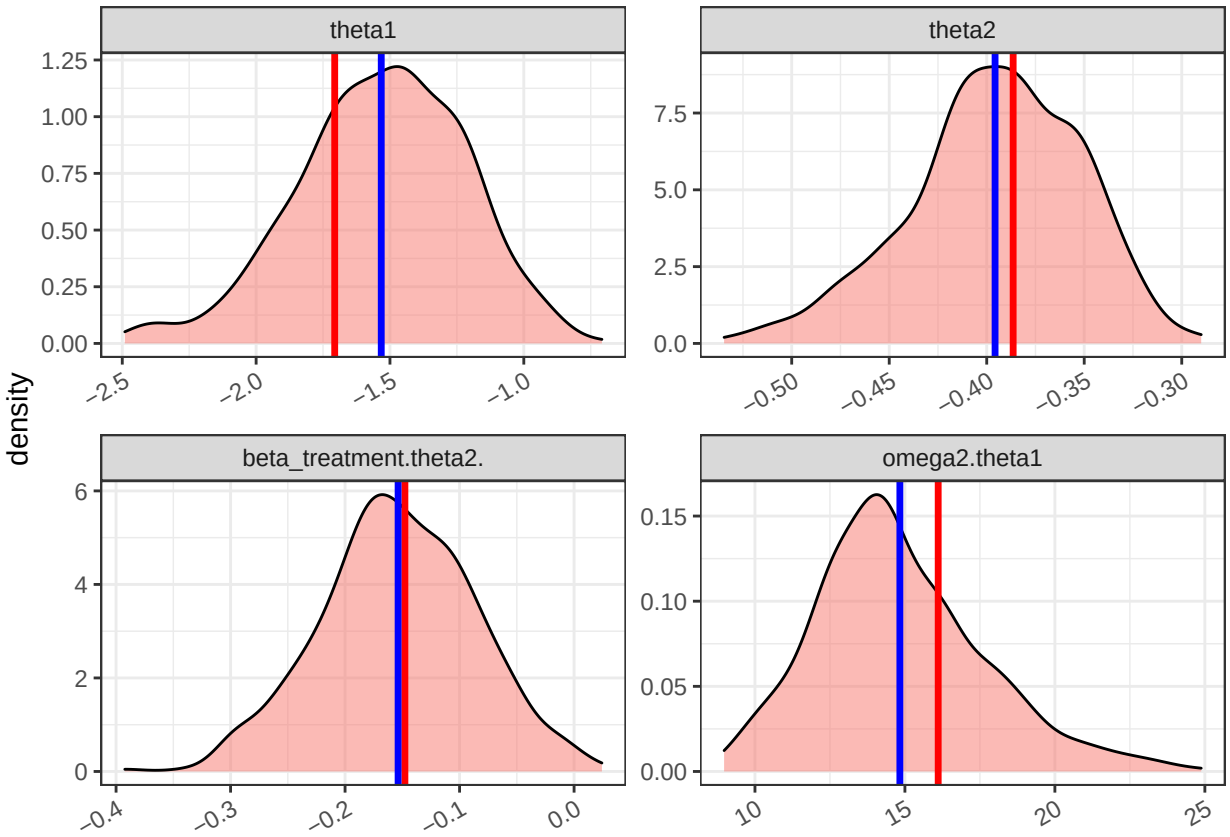
Conditional bootstrap We can also use conditional bootstrap, a non-parametric residual bootstrap which bootstraps samples from the conditional distributions and preserves the exact structure of the original dataset.

```
if(!runBootstrap)
  cond.bin <- read.table(file.path(saemixDir,"bootstrap","results","toenail_condBootstrap.res"), header=
    cond.bin <- saemix.bootstrap(binary.fit, method="conditional", nboot=nboot)
summary(cond.bin)
```

```
##    Replicate         theta1            theta2        beta_treatment.theta2.
##  Min.   :  1.0   Min.   :-2.4906   Min.   :-0.5347   Min.   :-0.39237
##  1st Qu.:125.8   1st Qu.:-1.7290   1st Qu.:-0.4202   1st Qu.:-0.19598
##  Median :250.5   Median :-1.5067   Median :-0.3928   Median :-0.15489
##  Mean   :250.5   Mean   :-1.5325   Mean   :-0.3957   Mean   :-0.15369
##  3rd Qu.:375.2   3rd Qu.:-1.2970   3rd Qu.:-0.3621   3rd Qu.:-0.10827
##  Max.   :500.0   Max.   :-0.7082   Max.   :-0.2900   Max.   : 0.02433
##  omega2.theta1
##  Min.   : 8.971
##  1st Qu.:12.851
##  Median :14.390
##  Mean   :14.831
##  3rd Qu.:16.433
##  Max.   :24.879
```

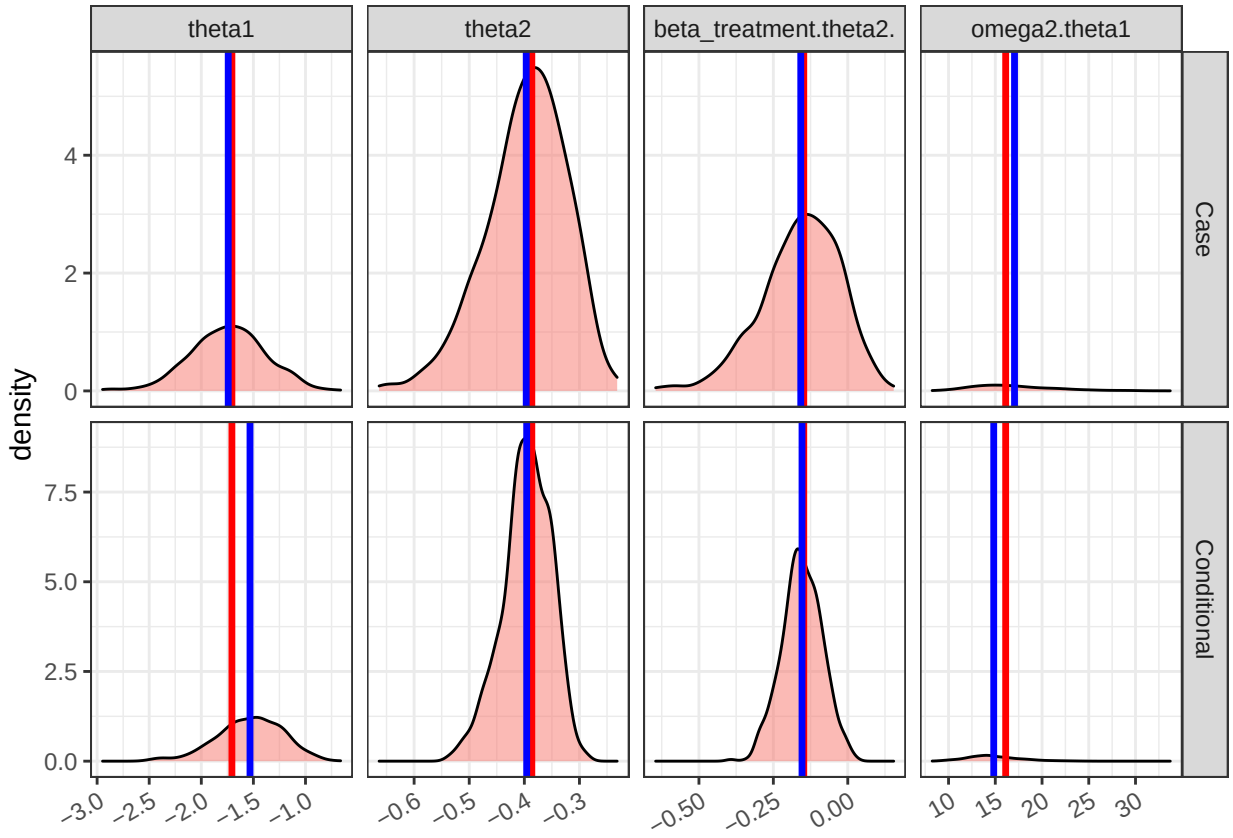
```
# Bootstrap distributions
#if(nboot<200) cat("The number of bootstrap samples is too low to provide good estimates of the confide
  resboot1<-cond.bin
```

```
ypd2<-NULL
for(icol in 1:4) {
  ypd2<-rbind(ypd2,data.frame(rep=resboot1[,1],Param=colnames(resboot1)[(icol+1)],value=resboot1[,(ic
}

ypd2$Param<-factor(ypd2$Param, levels = unique(ypd2$Param))
ypd2.fix<-ypd2[ypd2$Param %in% unique(ypd2$Param)[1:3],]
ypd2.iiv<-ypd2[ypd2$Param %in% unique(ypd2$Param)[4],]
ypd <- rbind(ypd,ypd2)

par.estim<-c(binary.fit@results@fixed.effects,diag(binary.fit@results@omega)[binary.fit@results@indx.
mean.bootDist<-apply(resboot1, 2, mean)[-c(1)]
df2<-data.frame(Param=unique(ypd2$Param), mean.boot=mean.bootDist, est.saemix=par.estim, Bootstrap="C
df<-rbind(df,df2)

  plot.density2<-ggplot(data=ypd2) + geom_density(aes(value,fill="red4"), alpha=0.5) +
  geom_vline(data=df2,aes(xintercept=est.saemix),colour="red",size=1.2) +
  geom_vline(data=df2,aes(xintercept=mean.boot),colour="blue",size=1.2) +
  theme_bw() + theme(axis.title.x = element_blank(),axis.text.x = element_text(size=9, angle=30, hjus
  facet_wrap(~Param, ncol=2, scales = 'free')
print(plot.density2)
```




```
plot.density3<-ggplot(data=ypd) + geom_density(aes(value,fill="red4"), alpha=0.5) +
  geom_vline(data=df,aes(xintercept=est.saemix),colour="red",size=1.2) +
  geom_vline(data=df,aes(xintercept=mean.boot),colour="blue",size=1.2) +
  theme_bw() + theme(axis.title.x = element_blank(),axis.text.x = element_text(size=9, angle=30, hjus
  facet_grid(Bootstrap~Param, scales = 'free')
```

```
#    facet_wrap(Bootstrap~Param, nrow=2, scales = 'free')

  print(plot.density3)
```




```
#}
```

Bootstrap results Here we produce a table showing tbe parameters estimated on the original dataset along with the bootstrap estimates (mean (SD) of the bootstrap distribution) and the 95% CI. The table is produced when the number of bootstrap is higher than 200 in the code below.

```
if(nboot<200) cat("The number of bootstrap samples is too low to provide good estimates of the confiden
  par.estim<-c(binary.fit@results@fixed.effects,diag(binary.fit@results@omega)[binary.fit@results@indx.
  df2<-data.frame(parameter=colnames(case.bin)[-c(1)], saemix=par.estim)
  for(i in 1:2) {
    if(i==1) {
      resboot1<-case.bin
      namboot<-"case"
      } else {
      resboot1<-cond.bin
      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  cNP.estimate
## 1                 theta1 -1.7063992 -1.74 (0.360) [-2.44, -1.077] -1.53 (0.312)
## 2                 theta2 -0.3864426 -0.40 (0.073) [-0.56, -0.279] -0.40 (0.044)
## 3 beta_treatment.theta2. -0.1477510 -0.16 (0.132)  [-0.44, 0.066] -0.15 (0.067)
## 4          omega2.theta1 16.1090732 17.06 (4.500) [10.36, 28.229] 14.83 (2.816)
##           cNP.CI
## 1 [-2.20, -0.98]
## 2 [-0.49, -0.32]
## 3 [-0.29, -0.02]
## 4 [10.01, 21.50]
```

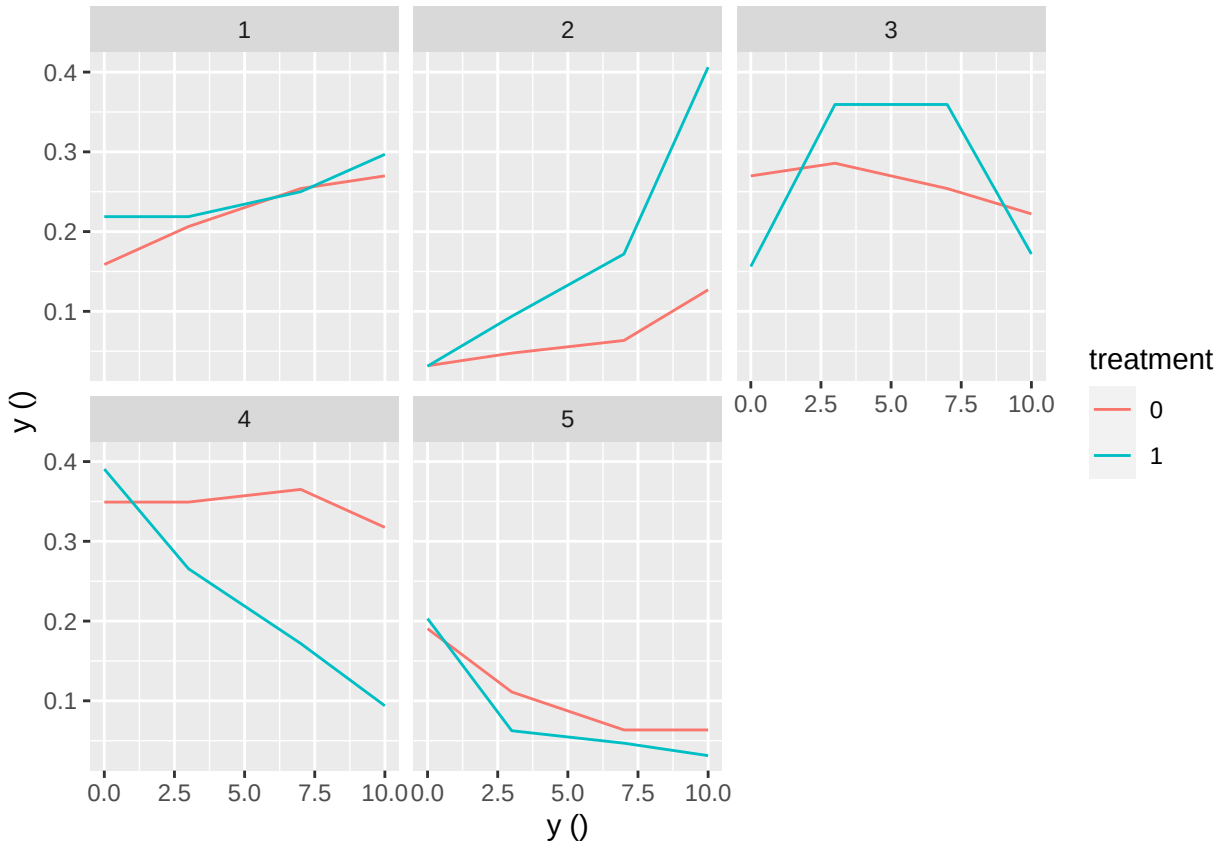
Categorical response model

Data The *knee.saemix* data represents pain scores recorded in a clinical study in 127 patients with sport related injuries treated with two different therapies. The pain occuring during knee movement was observed after 3,7 and 10 days of treatment. It was taken from the {catdata} package in R~[?] (dataset {knee}) and reformatted as follows

- a time column was added representing the day of the measurement (with 0 being the baseline value) and each observation corresponds to a different line in the dataset
- treatment was recoded as 0/1 (placebo/treatment), gender as 0/1 (male/female)
- {Age2} represents the squared of centered Age.

We can visualise the evolution of the proportion of each score over time using the *plotDiscreteData()* function with the outcome set to *categorical*, stratifying by treatment.

```
data(knee.saemix)

# 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")), verbose=FALSE)

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

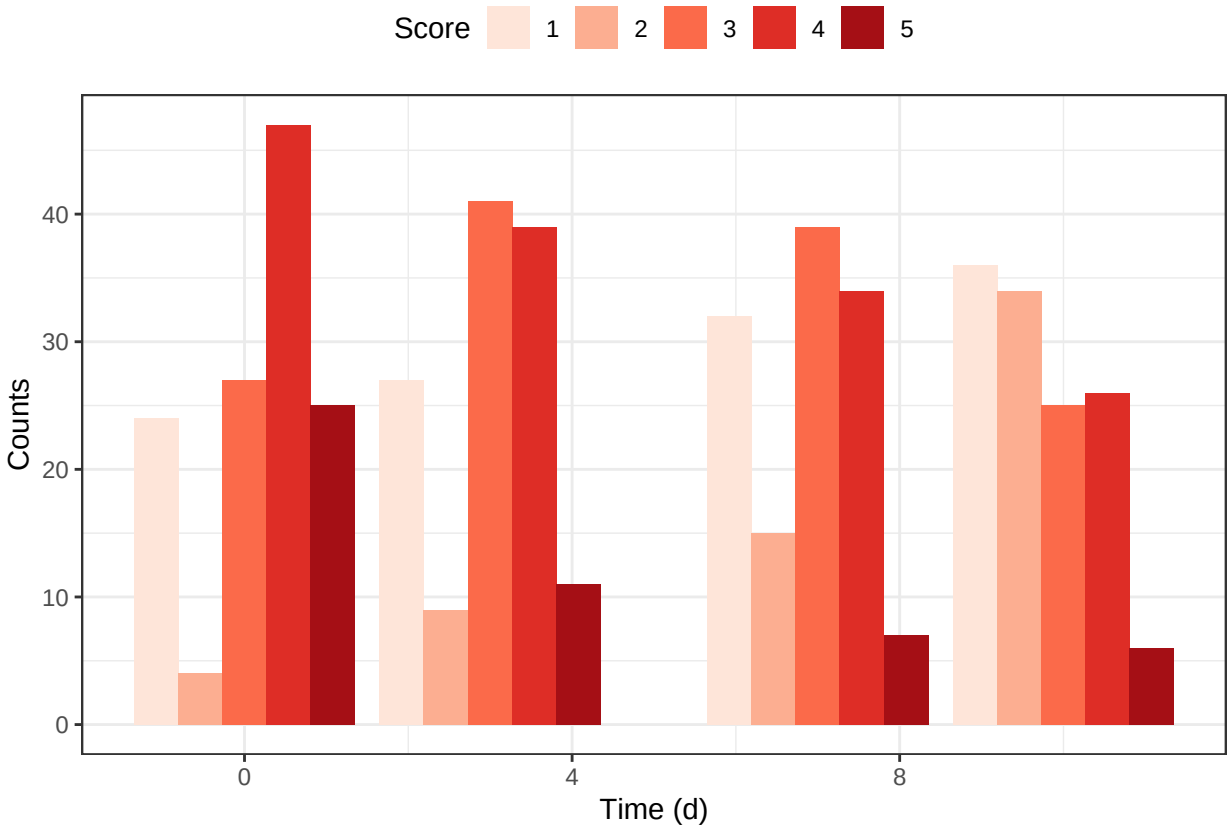


The following R code represents the data as barplots of the different pain scores as a function of time in study, illustrating a recovery as the proportion of lower pain scores increases.

```
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 The dataset is part of the datasets analysed in (Tutz 2012) with various methods described in the vignettes in the documentation of the *knee* dataset, but mainly as logistic regression on the response after 10 days, or as mixed binary regression after dichotomising the response. Here, we fit a proportional odds model to the full data. The probability $p_{ij} = P(Y_{ij} = 1|\theta_{1,i}, \theta_{2,i})$ associated with an event Y_{ij} at time t_{ij} is given by the following equation for the logit:

$$\begin{aligned} logit(P(Y_{ij} = 1|\psi_i)) &= \theta_{1,i} + \beta_i t_{ij} \\ logit(P(Y_{ij} = 2|\psi_i)) &= \theta_{1,i} + \theta_2 \\ logit(P(Y_{ij} = 3|\psi_i)) &= \theta_{1,i} + \theta_2 + \theta_3 \\ logit(P(Y_{ij} = 4|\psi_i)) &= \theta_{1,i} + \theta_2 + \theta_3 + \theta_4 \\ P(Y_{ij} = 4|\psi_i) &= 1 - \sum_k = 1^4 P(Y_{ij} = k|\psi_i) \end{aligned} \tag{1}$$

where θ_1 and β are assumed to have interindividual variability and to follow a normal distribution. β is the effect of time, θ_1 is the probability of a pain score of 1 and the other parameters represent an incremental risk to move into the higher pain category.

The following segment of code defines the ordinal model, computing the different logits for the different categories and deriving the corresponding probability given the observed data passed in *xidep*. We first fit a base model without covariate.

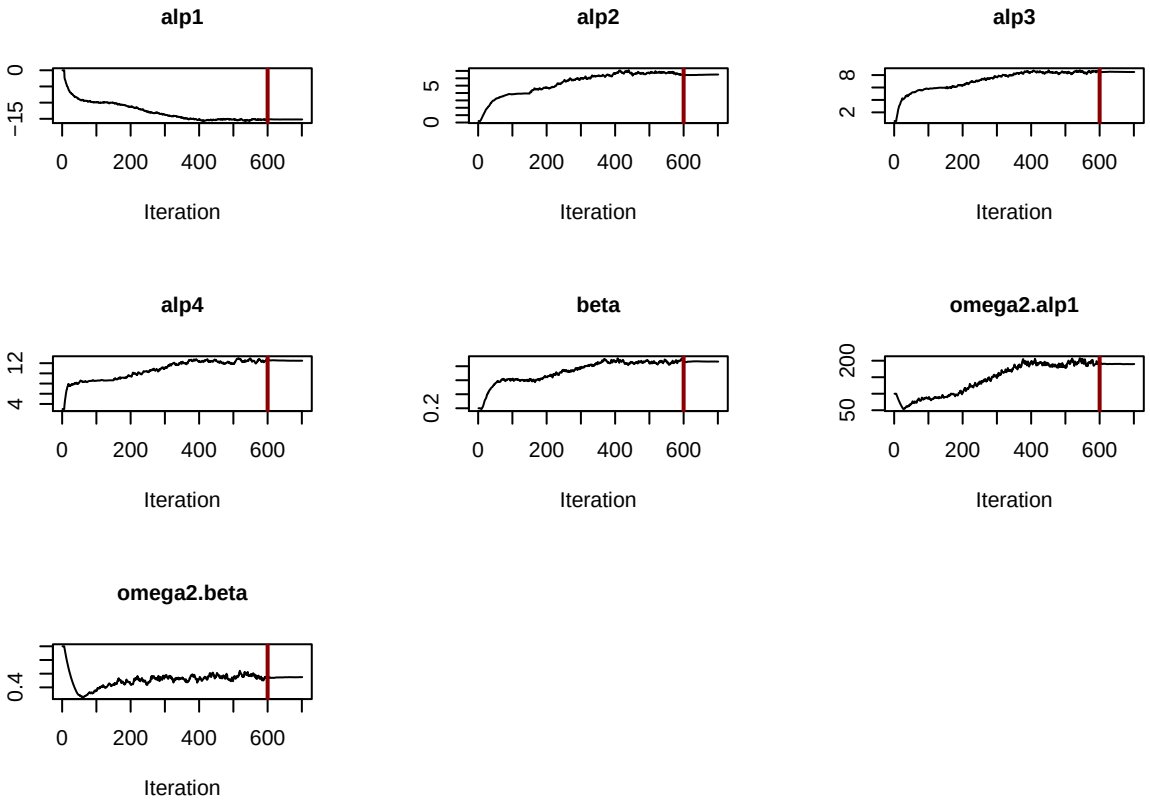
```
# Model for ordinal responses
ordinal.model<-function(psi,id,xidep) {
  y<-xidep[,1]
  time<-xidep[,2]
  alp1<-psi[id,1]
  alp2<-psi[id,2]
```

```
  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)
}
# simulate function
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)
}

# Saemix model
saemix.model<-saemixModel(model=ordinal.model,description="Ordinal categorical model",modeltype="likeli
                simulate.function=simulateOrdinal, 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(c(100, 1, 1, 1, 1)), covariance.model = diag(c(1,0,0,0,1)), verbose=FALS

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

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

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

```
## -----------------  Fixed effects  -----------------
## ---------------------------------------------------
##   Parameter Estimate
## 1      alp1   -15.21
## 2      alp2     6.51
## 3      alp3     8.49
## 4      alp4    12.48
## 5      beta     0.87
## ---------------------------------------------------
## -----------  Variance of random effects  -----------
## ---------------------------------------------------
##        Parameter Estimate
## alp1 omega2.alp1   189.79
## beta omega2.beta     0.55
## ---------------------------------------------------
## ------  Correlation matrix of random effects  ------
## ---------------------------------------------------
##             omega2.alp1 omega2.beta
## omega2.alp1 1.00        0.00
## omega2.beta 0.00        1.00
## ---------------------------------------------------
## --------------  Statistical criteria  -------------
## ---------------------------------------------------
##
## Likelihood computed by importance sampling
##       -2LL= 859.7992
##       AIC = 875.7992
##       BIC = 898.5527
## ---------------------------------------------------
```

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




```
## Note: comparable estimates obtained with Monolix (not same, but within CI)
## quite a lot of sensitivity to distributions (when using eg normal distributions in Monolix the param
```

We can then fit different models. Here we considered covariates on the two parameters with interindividual variability, first testing all covariates then reducing to a model with Age2 influencing θ_1 and treatment affecting the slope β, which had the lowest BICc, as shown using the *compare.saemix()* function.

```
# 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.cov1<-saemixModel(model=ordinal.model,description="Ordinal categorical model",modeltype="li
                               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, verbose=FALSE)
saemix.model.cov2<-saemixModel(model=ordinal.model,description="Ordinal categorical model",modeltype="li
                                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, verbose=FALSE)

ord.fit.cov1<-saemix(saemix.model.cov1,saemix.data,saemix.options)
ord.fit.cov2<-saemix(saemix.model.cov2,saemix.data,saemix.options)
BIC(ord.fit)
```

```
## [1] 898.5527
```

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

```
## [1] 912.2451
```

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

```
## [1] 886.9927
```

```
summary(ord.fit.cov2)
```

```
## ----------------------------------------------------
## -----------------  Fixed effects  ------------------
## ----------------------------------------------------
##              Parameter Estimate
## 1                  alp1   -20.19
## 2      beta_Age2(alp1)     0.05
## 3                  alp2     6.93
## 4                  alp3     8.70
## 5                  alp4    12.05
## 6                  beta     0.65
## 7 beta_treatment(beta)     0.56
## ----------------------------------------------------
## -----------  Variance of random effects  -----------
## ----------------------------------------------------
##        Parameter Estimate
## alp1 omega2.alp1   174.89
## beta omega2.beta     0.49
## ----------------------------------------------------
## ------  Correlation matrix of random effects  ------
## ----------------------------------------------------
##             omega2.alp1 omega2.beta
## omega2.alp1 1.00        0.00
## omega2.beta 0.00        1.00
## ----------------------------------------------------
## ---------------  Statistical criteria  -------------
## ----------------------------------------------------
##
## Likelihood computed by importance sampling
##       -2LL= 838.5508
##       AIC = 858.5508
##       BIC = 886.9927
## ----------------------------------------------------
```

```
# 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)
```

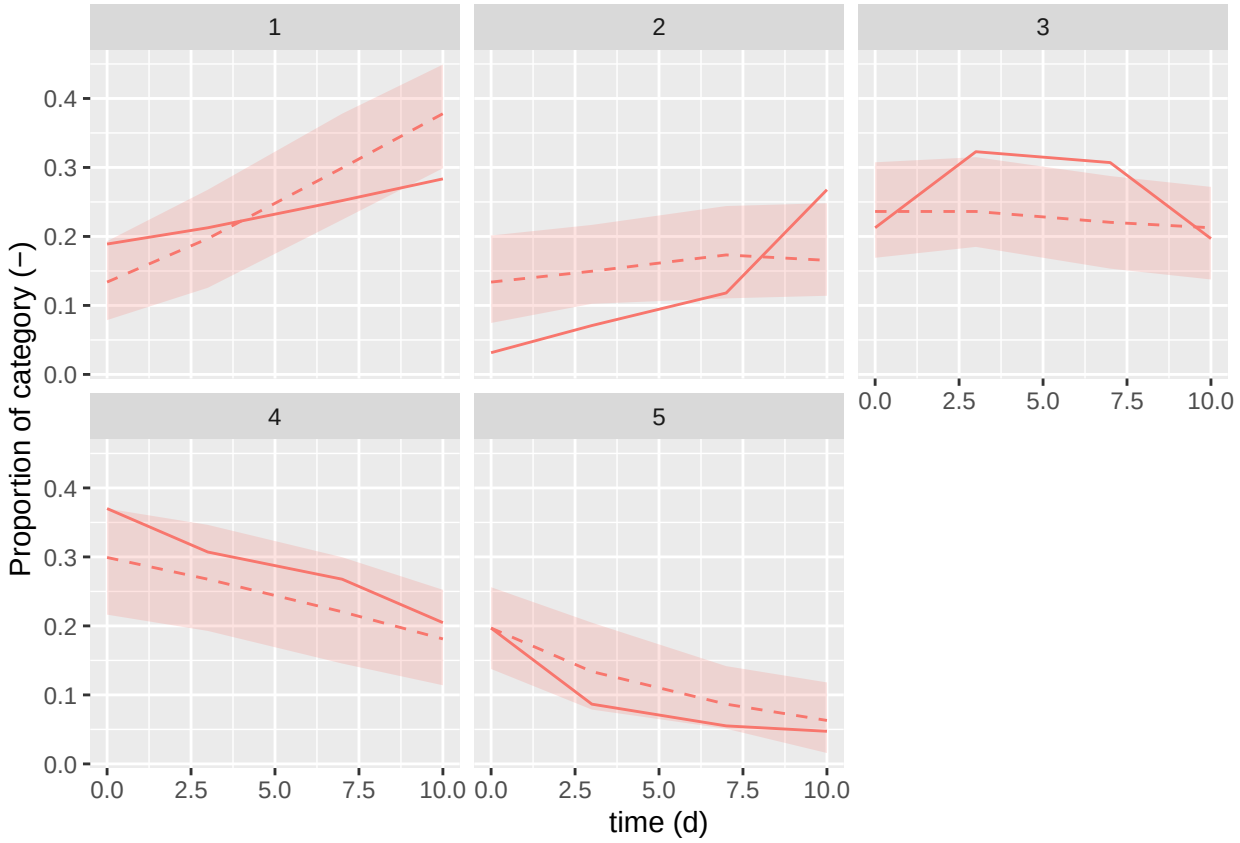
```
## Likelihoods calculated by importance sampling
```

```
##        AIC      BIC  BIC.cov
## 1 875.7992 898.5527 888.1790
## 2 866.7381 912.2451 901.8714
## 3 858.5508 886.9927 876.6190
```

Model evaluation In the code below we define a simulation function for the ordinal model and we apply it to the covariate model. The VPC show some model misspecification, especially in the intermediate pain scores, as well as a tendency to overestimate the improvement, driven by the reduction in the highest pain score. This

suggests the impact of time and treatment are not well taken into account in the current model.

```
### Simulations for VPC
nsim<-100
yfit<-ord.fit.cov2
yfit<-simulateDiscreteSaemix(yfit, nsim=nsim)
discreteVPC(yfit, outcome="categorical")
```



We can also look at the VPC for the median score in each treatment group to find that the model tends to underpredict the pain scores, especially in the group receiving the first therapy.

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

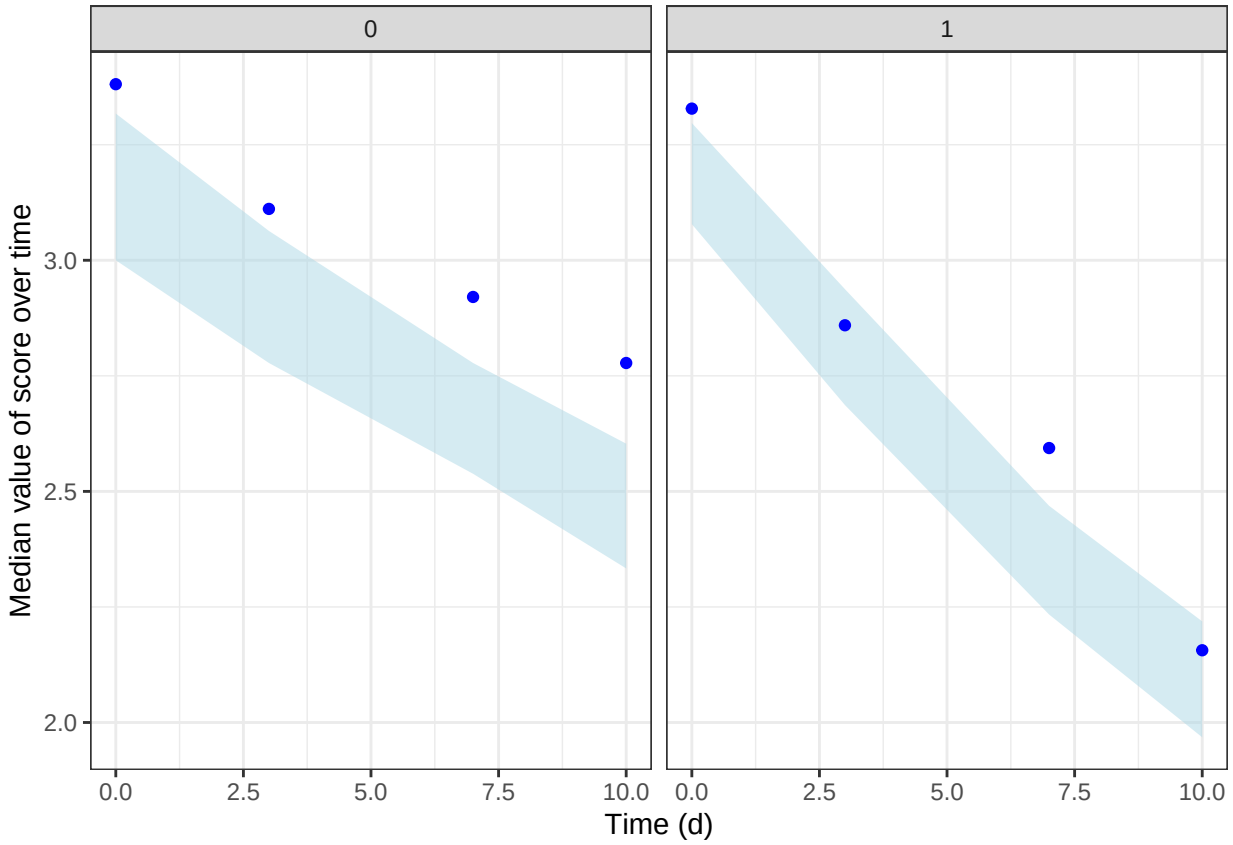
```
## `summarise()` has grouped output by 'time'. You can override using the `.groups` argument.
```

```
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) %>%
    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(saveFigs) {
  namfig<-"knee_medianScoreVPC.eps"
  cairo_ps(file = file.path(figDir, namfig), onefile = TRUE, fallback_resolution = 600, height=8.27, wi
  plot(kneeMedvpc)
  dev.off()
}
```

Estimation errors

Boostrap methods As previously, we can assess parameters uncertainty using bootstrap approaches. Here we load the results from the two bootstrap files prepared beforehand by running the *saemix.bootstrap* code with 500

simulations. 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.

```
if(runBootstrap) {
  case.ordinal <- saemix.bootstrap(ord.fit, method="case", nboot=nboot)
  cond.ordinal <- saemix.bootstrap(ord.fit, method="conditional", nboot=nboot)
} else {
 case.ordinal <- read.table(file.path(saemixDir,"bootstrap","results","knee_caseBootstrap.res"), header=
 cond.ordinal <- read.table(file.path(saemixDir,"bootstrap","results","knee_condBootstrap.res"), header=
 nboot<-dim(case.ordinal)[1]
}
case.ordinal <- case.ordinal[!is.na(case.ordinal[,2]),]
cond.ordinal <- cond.ordinal[!is.na(cond.ordinal[,2]),]

par.estim<-c(ord.fit@results@fixed.effects,diag(ord.fit@results@omega)[ord.fit@results@indx.omega])
df2<-data.frame(parameter=colnames(case.ordinal)[-c(1)], saemix=par.estim)
for(i in 1:2) {
  if(i==1) {
    resboot1<-case.ordinal
    namboot<-"case"
  } else {
    resboot1<-cond.ordinal
    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   cNP.estimate
## 1        alp1 -15.2065736  -16.12 (2.28) [-20.80, -11.70]  -14.68 (2.01)
## 2        alp2   6.5090520    7.07 (1.01)   [  5.25, 9.22]    5.97 (0.90)
## 3        alp3   8.4909399    8.96 (1.38)  [  6.62, 11.87]    8.40 (1.06)
## 4        alp4  12.4787329   13.26 (2.42)  [  9.46, 18.64]   12.86 (1.91)
## 5        beta   0.8662094    0.91 (0.11)   [  0.70, 1.14]    0.82 (0.11)
## 6 omega2.alp1 189.7883132 221.01 (64.76) [117.37, 363.58] 177.78 (48.53)
## 7 omega2.beta   0.5493485    0.57 (0.18)   [  0.26, 0.97]    0.52 (0.16)
##             cNP.CI
## 1 [-18.55, -11.09]
## 2   [  4.49, 7.77]
## 3  [  6.67, 10.69]
## 4  [  9.85, 17.34]
## 5   [  0.63, 1.03]
## 6 [ 99.82, 289.20]
## 7   [  0.25, 0.86]
```

Exact FIM by AGQ (code by Sebastian Ueckert)

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.

Here we use code provided by Sebastian Ueckert implementing the MC/AGQ approach, as the MCMC requires the installation of rStan. In this approach, the information matrix (FIM) over the population is first decomposed the sum of the individual FIM:

$$FIM(\Psi, \Xi) = \sum_{i=1}^{N} FIM(\Psi, \xi_i)$$

where ξ_i denotes the individual design in subject i. Assuming Q different elementary designs, the FIM can also be summed over the different designs weighted by the number of subjects N_q in design q as:

$$FIM(\Psi, \Xi) = \sum_{q=1}^{Q} N_q FIM(\Psi, \xi_q)$$

In the following, we first load the functions needed to compute the exact FIM. We then define a model object with the following components:

- *parameter_function*: a function returning the list of parameters as the combination of fixed and random effects
- *log_likelihood_function*: using the parameters, computes the log-likelihood for all y in the dataset
- *simulation_function*: using the parameters, computes the log-likelihood and produces a random sample from the corresponding distribution
- *inverse_simulation_function*: supposed to be the quantile function but not quite sure :-/ (here, returns the category in which is urand)
- *mu*: the fixed parameters
- *omega*: the variance-covariance matrix

For *mu* and *omega*, we use the results from the saemix fit. Here we show the computation for the model without covariates. For a model with covariates, we would need to compute the FIM for each combination of covariates for categorical covariates, or for each subject with continuous covariates like Age2 here.

```
# Code Sebastian
source(file.path(dirAGQ,"default_settings.R"))
source(file.path(dirAGQ,"helper_functions.R"))
source(file.path(dirAGQ,"integration.R"))
source(file.path(dirAGQ,"model.R"))

saemix.fit <- ord.fit

# Setting up ordinal model
model <- Model$new(
  parameter_function = function(mu, b) list(alp1=mu[1]+b[1], alp2=mu[2], alp3=mu[3], alp4=mu[4], beta=mu
  log_likelihood_function = function(y, design, alp1, alp2, alp3, alp4, beta) {
    logit1<-alp1 + beta*design$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)
    log(pobs)
  },
```

```
  simulation_function = function(design, alp1, alp2, alp3, alp4, beta) {
    logit1<-alp1 + beta*design$time
    logit2<-logit1+alp2
    logit3<-logit2+alp3
    logit4<-logit3+alp4
    pge1<-exp(logit1)/(1+exp(logit1))
    pge2<-exp(logit2)/(1+exp(logit2))
    pge3<-exp(logit3)/(1+exp(logit3))
    pge4<-exp(logit4)/(1+exp(logit4))
    x<-runif(length(time))
    ysim<-1+as.integer(x>pge1)+as.integer(x>pge2)+as.integer(x>pge3)+as.integer(x>pge4)
  },
  inverse_simulation_function = function(design, urand,alp1, alp2, alp3, alp4, beta) {
    if(is.null(urand)) return(seq_along(design$time))
    logit1<-alp1 + beta*design$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))
    1+as.integer(urand>pge1)+as.integer(urand>pge2)+as.integer(urand>pge3)+as.integer(urand>pge4)
  },
  mu = saemix.fit@results@fixed.effects,
  omega = saemix.fit@results@omega[c(1,5),c(1,5)]
  )


# define settings (agq with 3 grid points, quasi random monte-carlo and 500 samples)
settings <- defaults.agq(gq.quad_points = 3,  y_integration.method = "qrmc", y_integration.n_samples = 5

#### Design
# Checking whether everyone has the same visits - yes
time.patterns<-tapply(knee.saemix$time, knee.saemix$id, function(x) paste(x,collapse="-"))
unique(time.patterns)
```

```
##         1
## "0-3-7-10"
```

```
# same 4 times for all subjects (0, 3, 7, 10)
design <- data.frame(time=sort(unique(knee.saemix$time)))
fim <- length(unique(knee.saemix$id)) * calc_fim(model, design, settings)
print(fim)
```

```
##                    mu1          mu2         mu3         mu4          mu5
## mu1       8.806568e-01  0.743810295  0.61930629  0.21515836   1.99179508
## mu2       7.438103e-01  3.310200184  0.60722170  0.31107714  -2.99491659
## mu3       6.193063e-01  0.607221702  2.16029402  0.24018578  -0.09697394
## mu4       2.151584e-01  0.311077138  0.24018578  0.47508343  -0.04036202
## mu5       1.991795e+00 -2.994916590 -0.09697394 -0.04036202 112.40335959
## omega1.1 -6.549488e-05 -0.005795345 -0.00961219 -0.00706586  -0.04527356
## omega2.2  1.003040e+00 -5.449476775 -3.91635253 -1.68817482  -1.37022748
##               omega1.1    omega2.2
```

```
## mu1      -6.549488e-05  1.00304000
## mu2      -5.795345e-03 -5.44947677
## mu3      -9.612190e-03 -3.91635253
## mu4      -7.065860e-03 -1.68817482
## mu5      -4.527356e-02 -1.37022748
## omega1.1  7.245165e-04 -0.02596371
## omega2.2 -2.596371e-02 87.73819946
```

```
# calculate rse
rse <- calc_rse(model, fim)
print(rse)
```

```
##       mu1       mu2       mu3       mu4       mu5  omega1.1  omega2.2
## -13.52621  12.53514  11.85812  16.33464  13.66811  28.73448  31.39889
```

```
est.se<-sqrt(diag(solve(fim)))
df <- data.frame(param=c(model$mu,diag(model$omega)),se=est.se)
df$rse <- abs(df$se/df$param*100)

print(df)
```

```
##                param         se      rse
## mu1      -15.2065736  2.0568723 13.52621
## mu2        6.5090520  0.8159188 12.53514
## mu3        8.4909399  1.0068659 11.85812
## mu4       12.4787329  2.0383562 16.33464
## mu5        0.8662094  0.1183945 13.66811
## omega1.1 189.7883132 54.5346783 28.73448
## omega2.2   0.5493485  0.1724893 31.39889
```

Comparing the SE with the different approaches

```
# Adding the exact FIM estimates to df2
l1<-paste0(format(par.estim, digits=2)," (",format(est.se,digits=2, trim=T),")")
ci.low <- par.estim - 1.96*est.se
ci.up <- par.estim + 1.96*est.se
l2<-paste0("[",format(ci.low, digits=2),", ",format(ci.up,digits=2, trim=T),"]")
df2<-cbind(df2, l1, l2)
colnames(df2)[7:8]<-paste0("FIM.",c("estimate","CI"))
print(df2)
```

```
##     parameter      saemix  case.estimate          case.CI   cNP.estimate
## 1        alp1 -15.2065736  -16.12 (2.28) [-20.80, -11.70]  -14.68 (2.01)
## 2        alp2   6.5090520    7.07 (1.01)   [  5.25, 9.22]    5.97 (0.90)
## 3        alp3   8.4909399    8.96 (1.38)  [  6.62, 11.87]    8.40 (1.06)
## 4        alp4  12.4787329   13.26 (2.42)  [  9.46, 18.64]   12.86 (1.91)
## 5        beta   0.8662094    0.91 (0.11)   [  0.70, 1.14]    0.82 (0.11)
## 6 omega2.alp1 189.7883132 221.01 (64.76) [117.37, 363.58] 177.78 (48.53)
## 7 omega2.beta   0.5493485    0.57 (0.18)   [  0.26, 0.97]    0.52 (0.16)
##             cNP.CI   FIM.estimate          FIM.CI
## 1 [-18.55, -11.09]  -15.21 (2.06) [-19.24, -11.18]
## 2   [  4.49, 7.77]    6.51 (0.82)   [  4.91, 8.11]
## 3  [  6.67, 10.69]    8.49 (1.01)  [  6.52, 10.46]
## 4  [  9.85, 17.34]   12.48 (2.04)  [  8.48, 16.47]
## 5   [  0.63, 1.03]    0.87 (0.12)   [  0.63, 1.10]
## 6 [ 99.82, 289.20] 189.79 (54.53) [ 82.90, 296.68]
## 7   [  0.25, 0.86]    0.55 (0.17)   [  0.21, 0.89]
```

References

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

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