Saemix 3 - comparing the bootstrap and expected FIM by MC/AGQ for categorical models

Emmanuelle

06/2023

Version

Use saemix version $\geq$ 3.2

Objective

Compare the bootstrap estimates of the SE from **saemix** to the expected SE predicted for the categorical data example *knee.saemix* (see main notebook), which we will compute using a numerical computation proposed by Ueckert et al. (2016).

The main analysis of the *knee.saemix* dataset can be found in the main notebook.

Rationale For non-Gaussian models, the FOCE approximation used in **saemix** is poor, and the exact FIM should be computed. Because of the integration however, this computation is computationally intensive and has not yet been implemented in the package (work in progress). However, a similar issue arises in optimal design where computing the *expected* FIM is used to evaluate and optimise designs.

In the context of optimal design, two approaches have been proposed using either numerical integration by a combination eof 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 will show how to use the MC/AGQ method to obtain the *expected* FIM.

Because the dataset that we are using is homogenous (all patients have the same number of measurements at the same times), and we have a relatively large number of subjects, we anticipate that the expected FIM will be close to the observed FIM, allowing us to assess whether the bootstrap estimates are reasonable.

Setting up the libraries and paths

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 (Schauberger and Tutz 2020) (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.

```
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)
```

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

Estimation errors

Boostrap methods 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 predicted FIM by AGQ (code by Sebastian Ueckert)

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       0.8756643073  0.814997601  0.67350019  0.210948860  2.11551121
```

```
## mu2      0.8149976014  3.075649531  0.62317891  0.210334136 -1.47839784
## mu3      0.6735001934  0.623178908  1.99009458  0.326424245 -0.32197612
## mu4      0.2109488603  0.210334136  0.32642424  0.637148551 -1.08910639
## mu5      2.1155112118 -1.478397841 -0.32197612 -1.089106387 93.24206363
## omega1.1 0.0001668344 -0.007266133 -0.01136008 -0.007509462 -0.03793081
## omega2.2 0.1852513140 -3.894028842 -3.13237842 -2.139321938  6.64065807
##               omega1.1     omega2.2
## mu1       0.0001668344  0.185251314
## mu2      -0.0072661329 -3.894028842
## mu3      -0.0113600807 -3.132378421
## mu4      -0.0075094617 -2.139321938
## mu5      -0.0379308107  6.640658068
## omega1.1  0.0007331782 -0.008616313
## omega2.2 -0.0086163134 65.851316059
```

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

```
##       mu1       mu2       mu3       mu4       mu5  omega1.1  omega2.2
## -13.89234  13.06156  12.70020  14.28702  15.30587  28.52350  31.91025
```

```
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  2.1125490 13.89234
## mu2        6.5090520  0.8501840 13.06156
## mu3        8.4909399  1.0783665 12.70020
## mu4       12.4787329  1.7828387 14.28702
## mu5        0.8662094  0.1325809 15.30587
## omega1.1 189.7883132 54.1342739 28.52350
## omega2.2   0.5493485  0.1752985 31.91025
```

Work in progress Access directly the functions computing the individual FIM and feed them the individual observations to compute the observed FIM instead of simulating observations to obtain the expected FIM.

Comparing the SE with the different approaches The expected SE are indeed very similar to the ones obtained by the two bootstrap methods.

```
# 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 (2.11) [-19.35, -11.07]
## 2   [  4.49, 7.77]    6.51 (0.85)   [  4.84, 8.18]
## 3  [  6.67, 10.69]    8.49 (1.08)  [  6.38, 10.60]
## 4  [  9.85, 17.34]   12.48 (1.78)  [  8.98, 15.97]
## 5   [  0.63, 1.03]    0.87 (0.13)   [  0.61, 1.13]
## 6 [ 99.82, 289.20] 189.79 (54.13) [ 83.69, 295.89]
## 7   [  0.25, 0.86]    0.55 (0.18)   [  0.21, 0.89]
```

```
tab1 <- xtable(df2)
print.xtable(tab1, include.rownames = FALSE)
```

```
## % latex table generated in R 4.2.2 by xtable 1.8-4 package
## % Mon Jun 26 21:56:54 2023
## \begin{table}[ht]
## \centering
## \begin{tabular}{lrllllll}
##   \hline
## parameter & saemix & case.estimate & case.CI & cNP.estimate & cNP.CI & FIM.estimate & FIM.CI \\
##   \hline
## alp1 & -15.21 & -16.12 (2.28) & [-20.80, -11.70] & -14.68 (2.01) & [-18.55, -11.09] & -15.21 (2.11) &
##   alp2 & 6.51 &   7.07 (1.01) & [  5.25, 9.22] &   5.97 (0.90) & [  4.49, 7.77] &   6.51 (0.85) & [
##   alp3 & 8.49 &   8.96 (1.38) & [  6.62, 11.87] &   8.40 (1.06) & [  6.67, 10.69] &   8.49 (1.08) &
##   alp4 & 12.48 &  13.26 (2.42) & [  9.46, 18.64] &  12.86 (1.91) & [  9.85, 17.34] &  12.48 (1.78) &
##   beta & 0.87 &   0.91 (0.11) & [  0.70, 1.14] &   0.82 (0.11) & [  0.63, 1.03] &   0.87 (0.13) & [
##   omega2.alp1 & 189.79 & 221.01 (64.76) & [117.37, 363.58] & 177.78 (48.53) & [ 99.82, 289.20] & 189
##   omega2.beta & 0.55 &   0.57 (0.18) & [  0.26, 0.97] &   0.52 (0.16) & [  0.25, 0.86] &   0.55 (0.1
##    \hline
## \end{tabular}
## \end{table}
```

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

Schauberger G, **Tutz G** (2020). catdata: Categorical Data. R package version 1.2.2. https://CRAN.R-project.org/package=catdata