Conditional distributions

Emmanuelle Comets

10/06/2024

Objective

Compute conditional distributions

Setup

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

Continuous response model

Theophylline

```
data(theo.saemix)
saemix.data<-saemixData(name.data=theo.saemix,header=TRUE,sep=" ",na=NA,
  name.group=c("Id"),name.predictors=c("Dose","Time"),
  name.response=c("Concentration"),name.covariates=c("Weight","Sex"),
  units=list(x="hr",y="mg/L",covariates=c("kg","-")), name.X="Time")
```

```
##
##
## The following SaemixData object was successfully created:
##
## Object of class SaemixData
##     longitudinal data for use with the SAEM algorithm
## Dataset theo.saemix
##     Structured data: Concentration ~ Dose + Time | Id
##     X variable for graphs: Time (hr)
##     covariates: Weight (kg), Sex (-)
##       reference class for covariate Sex :  0
```

```
model1cpt<-function(psi,id,xidep) {
    dose<-xidep[,1]
    tim<-xidep[,2]
    ka<-psi[id,1]
    V<-psi[id,2]
    CL<-psi[id,3]
    k<-CL/V
```

```
    ypred<-dose*ka/(V*(ka-k))*(exp(-k*tim)-exp(-ka*tim))
    return(ypred)
}

# Model with covariates
saemix.model<-saemixModel(model=model1cpt,
                          description="One-compartment model with first-order absorption",
                          psi0=matrix(c(1.,20,0.5,0.1,0,-0.01),ncol=3,byrow=TRUE,
                                      dimnames=list(NULL, c("ka","V","CL"))),transform.par=c(1,1,1),
                          covariate.model=matrix(c(0,0,1,0,0,0),ncol=3,byrow=TRUE),fixed.estim=c(1,1,1)
                          covariance.model=matrix(c(1,0,0,0,1,1,0,1,1),ncol=3,byrow=TRUE),
                          omega.init=matrix(c(1,0,0,0,1,0,0,0,1),ncol=3,byrow=TRUE),error.model="combin
```

```
##
##
## The following SaemixModel object was successfully created:
##
## Nonlinear mixed-effects model
##   Model function:  One-compartment model with first-order absorption
##   Model type:  structural
## function(psi,id,xidep) {
##     dose<-xidep[,1]
##     tim<-xidep[,2]
##     ka<-psi[id,1]
##     V<-psi[id,2]
##     CL<-psi[id,3]
##     k<-CL/V
##     ypred<-dose*ka/(V*(ka-k))*(exp(-k*tim)-exp(-ka*tim))
##     return(ypred)
## }
##   Nb of parameters: 3
##       parameter names:  ka V CL
##       distribution:
##      Parameter Distribution Estimated
## [1,] ka        log-normal   Estimated
## [2,] V         log-normal   Estimated
## [3,] CL        log-normal   Estimated
##   Variance-covariance matrix:
##    ka V CL
## ka  1 0  0
## V   0 1  1
## CL  0 1  1
##   Error model: combined , initial values: a.1=1 b.1=1
##   Covariate model:
##      ka V CL
## [1,]  0 0  1
## [2,]  0 0  0
##     Initial values
##               ka  V    CL
## Pop.CondInit 1.0 20  0.50
## Cov.CondInit 0.1  0 -0.01
```

```
saemix.options<-list(seed=39546,save=FALSE,save.graphs=FALSE, displayProgress=FALSE)
saemix.fit<-saemix(saemix.model,saemix.data,saemix.options)
```

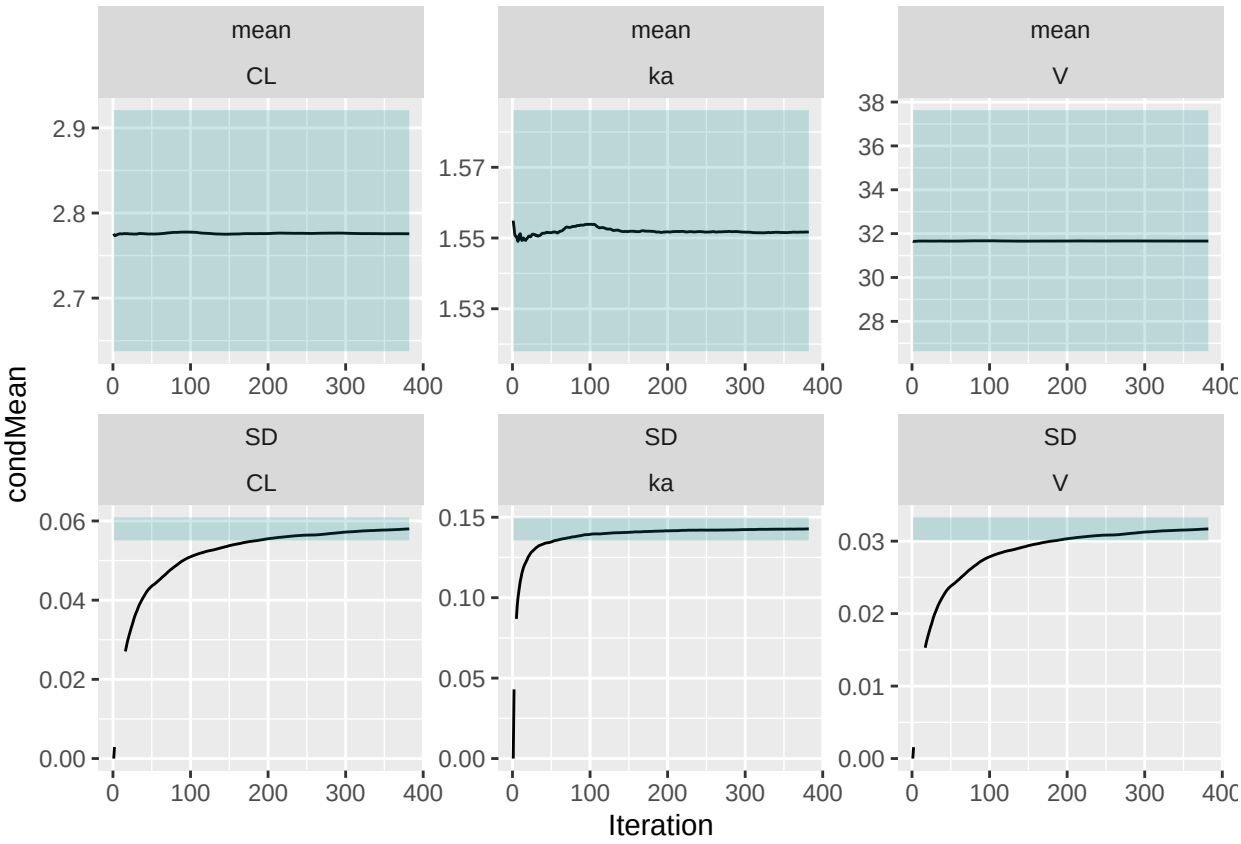
```
## Nonlinear mixed-effects model fit by the SAEM algorithm
## ----------------------------------
## ----          Data              ----
## ----------------------------------
## Object of class SaemixData
##     longitudinal data for use with the SAEM algorithm
## Dataset theo.saemix
##     Structured data: Concentration ~ Dose + Time | Id
##     X variable for graphs: Time (hr)
##     covariates: Weight (kg), Sex (-)
##       reference class for covariate Sex :  0
## Dataset characteristics:
##     number of subjects:     12
##     number of observations: 120
##     average/min/max nb obs: 10.00  /  10  /  10
## First 10 lines of data:
##    Id    Dose  Time Concentration Weight Sex mdv cens occ ytype
## 1   1 319.992  0.25          2.84   79.6   1   0    0   1     1
## 2   1 319.992  0.57          6.57   79.6   1   0    0   1     1
## 3   1 319.992  1.12         10.50   79.6   1   0    0   1     1
## 4   1 319.992  2.02          9.66   79.6   1   0    0   1     1
## 5   1 319.992  3.82          8.58   79.6   1   0    0   1     1
## 6   1 319.992  5.10          8.36   79.6   1   0    0   1     1
## 7   1 319.992  7.03          7.47   79.6   1   0    0   1     1
## 8   1 319.992  9.05          6.89   79.6   1   0    0   1     1
## 9   1 319.992 12.12          5.94   79.6   1   0    0   1     1
## 10  1 319.992 24.37          3.28   79.6   1   0    0   1     1
## ----------------------------------
## ----          Model             ----
## ----------------------------------
## Nonlinear mixed-effects model
##   Model function:  One-compartment model with first-order absorption
##   Model type:  structural
## function(psi,id,xidep) {
##     dose<-xidep[,1]
##     tim<-xidep[,2]
##     ka<-psi[id,1]
##     V<-psi[id,2]
##     CL<-psi[id,3]
##     k<-CL/V
##     ypred<-dose*ka/(V*(ka-k))*(exp(-k*tim)-exp(-ka*tim))
##     return(ypred)
## }
## <bytecode: 0x5650002e1898>
##   Nb of parameters: 3
##       parameter names:  ka V CL
##       distribution:
##      Parameter Distribution Estimated
## [1,] ka        log-normal   Estimated
## [2,] V         log-normal   Estimated
## [3,] CL        log-normal   Estimated
##   Variance-covariance matrix:
##    ka V CL
## ka  1 0  0
```

```
## V   0 1  1
## CL  0 1  1
##   Error model: combined , initial values: a.1=1 b.1=1
##   Covariate model:
##        [,1] [,2] [,3]
## Weight    0    0    1
##     Initial values
##               ka  V    CL
## Pop.CondInit 1.0 20  0.50
## Cov.CondInit 0.1  0 -0.01
## ----------------------------------
## ----    Key algorithm options  ----
## ----------------------------------
##     Estimation of individual parameters (MAP)
##     Estimation of standard errors and linearised log-likelihood
##     Estimation of log-likelihood by importance sampling
##     Number of iterations:  K1=300, K2=100
##     Number of chains:  5
##     Seed:  39546
##     Number of MCMC iterations for IS:  5000
##     Simulations:
##         nb of simulated datasets used for npde:  1000
##         nb of simulated datasets used for VPC:  100
##     Input/output
##         save the results to a file:  FALSE
##         save the graphs to files:  FALSE
## ----------------------------------------------------
## ----                  Results                   ----
## ----------------------------------------------------
## -----------------  Fixed effects  ------------------
## ----------------------------------------------------
##      Parameter       Estimate SE     CV(%) p-value
## [1,] ka               1.5565  0.3050 19.6  -
## [2,] V               31.6621  1.4946  4.7  -
## [3,] CL               4.4308  1.9206 43.3  -
## [4,] beta_Weight(CL) -0.0067  0.0061 91.3  0.27
## [5,] a.1              0.5734  0.0935 16.3  -
## [6,] b.1              0.0748  0.0223 29.9  -
## ----------------------------------------------------
## -----------  Variance of random effects  -----------
## ----------------------------------------------------
##       Parameter Estimate SE    CV(%)
## ka    omega2.ka 0.412    0.179 44
## V     omega2.V  0.019    0.011 56
## CL    omega2.CL 0.064    0.031 48
## covar cov.V.CL  0.035    0.016 45
## ----------------------------------------------------
## ------  Correlation matrix of random effects  ------
## ----------------------------------------------------
##           omega2.ka omega2.V omega2.CL
## omega2.ka 1         0        0
## omega2.V  0         1        1
## omega2.CL 0         1        1
## ----------------------------------------------------
```

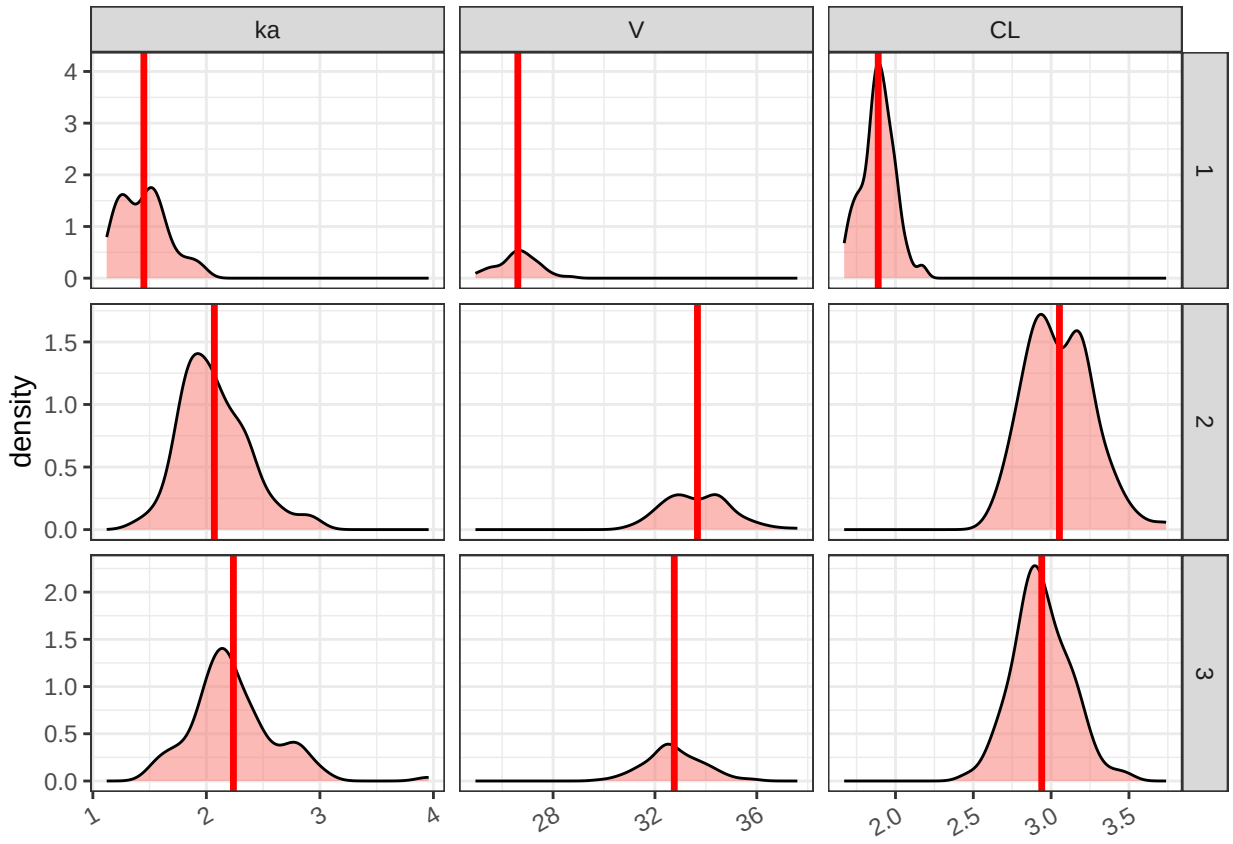
```
## ---------------  Statistical criteria  -------------
## -------------------------------------------------
## Likelihood computed by linearisation
##       -2LL= 330.7213
##       AIC = 350.7213
##       BIC = 355.5704
##
## Likelihood computed by importance sampling
##       -2LL= 333.9945
##       AIC = 353.9945
##       BIC = 358.8436
## -------------------------------------------------
```

```
# Conditional distribution
saemix.fit <- conddist.saemix(saemix.fit, nsamp=100, plot=TRUE)
```

```
## Warning in sqrt(varik): Production de NaN

## Warning in sqrt(varik): Production de NaN

## Warning in sqrt(varik): Production de NaN

## Warning in sqrt(varik): Production de NaN

## Warning in sqrt(varik): Production de NaN

## Warning in sqrt(varik): Production de NaN

## Warning in sqrt(varik): Production de NaN

## Warning in sqrt(varik): Production de NaN

## Warning in sqrt(varik): Production de NaN

## Warning in sqrt(varik): Production de NaN

## Warning in sqrt(varik): Production de NaN

## Warning in sqrt(varik): Production de NaN

## Warning in sqrt(varik): Production de NaN

## Warning in sqrt(varik): Production de NaN
```



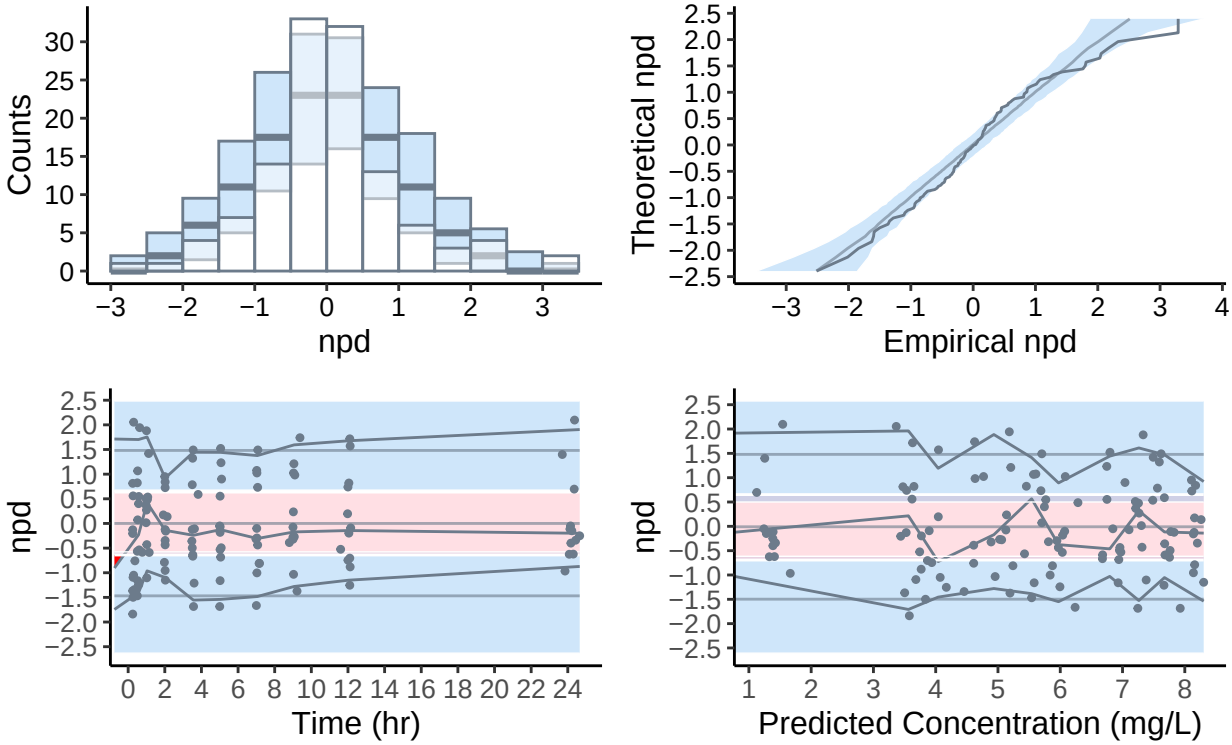

```
psi.samp <- saemix.fit@results@psi.samp
yplot <- NULL
for(isamp in 1:dim(psi.samp)[3]) {
    yplot <- rbind(yplot, data.frame(id=1:3, irep=isamp,psi.samp[1:3,,isamp]))
}
ypd2 <- rbind(cbind(yplot[,c(1:2)],value=yplot[,3],param=saemix.fit@model@name.fixed[1]),
              cbind(yplot[,c(1:2)],value=yplot[,4],param=saemix.fit@model@name.fixed[2]),
              cbind(yplot[,c(1:2)],value=yplot[,5],param=saemix.fit@model@name.fixed[3]))
df.lines <- NULL
for(i in 1:3) {
  df.lines <- rbind(df.lines,
                    data.frame(id=1:3,mean=tapply(yplot[,i+2],yplot$id, mean), median=tapply(yplot[,i+2]
}

ypd2$param<-factor(ypd2$param, levels = c("ka","V","CL"))
plot.density2<-ggplot(data=ypd2) + geom_density(aes(value,fill="red4"), alpha=0.5) +
    geom_vline(data=df.lines,aes(xintercept=mean),colour="red",linewidth=1.2) +
    theme_bw() + theme(axis.title.x = element_blank(),axis.text.x = element_text(size=9, angle=30, hjus
    facet_grid(id~factor(param), scales = 'free')
plot.density2
```




```
# using npde
ynpde<-npdeSaemix(saemix.fit)
```

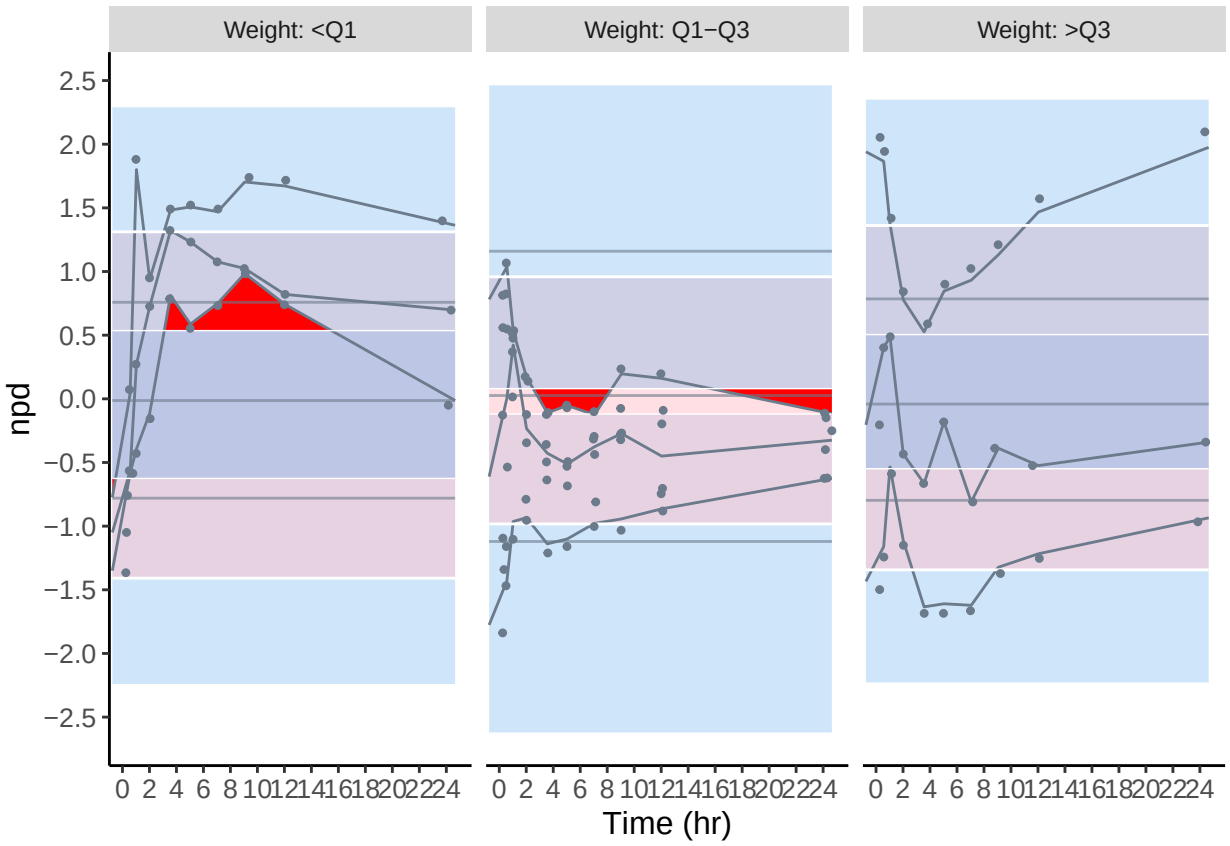
```
## Warning in read(x, dat, detect = detect, verbose = verbose): NAs introduits
## lors de la conversion automatique

## Warning in read(x, dat, detect = detect, verbose = verbose): NAs introduits
## lors de la conversion automatique

## Warning in read(x, dat, detect = detect, verbose = verbose): NAs introduits
## lors de la conversion automatique

## Warning in read(x, dat, detect = detect, verbose = verbose): NAs introduits
## lors de la conversion automatique

## Warning in read(x, dat, detect = detect, verbose = verbose): NAs introduits
## lors de la conversion automatique

## Warning in which(!is.na(as.integer(object@name.covariates))): NAs introduits
## lors de la conversion automatique

## ------------------------------------------------
## Distribution of npde :
##       nb of obs: 120
##            mean= 0.05215   (SE= 0.088 )
##        variance= 0.9338   (SE= 0.12 )
##        skewness= 0.5886
##        kurtosis= 1.539
```

```
## ------------------------------------------------
## Statistical tests (adjusted p-values):
##   t-test                 : 1
##   Fisher variance test   : 1
##   SW test of normality   : 0.00541 **
##   Global test            : 0.00541 **
## ---
## Signif. codes: '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1
## ------------------------------------------------
```

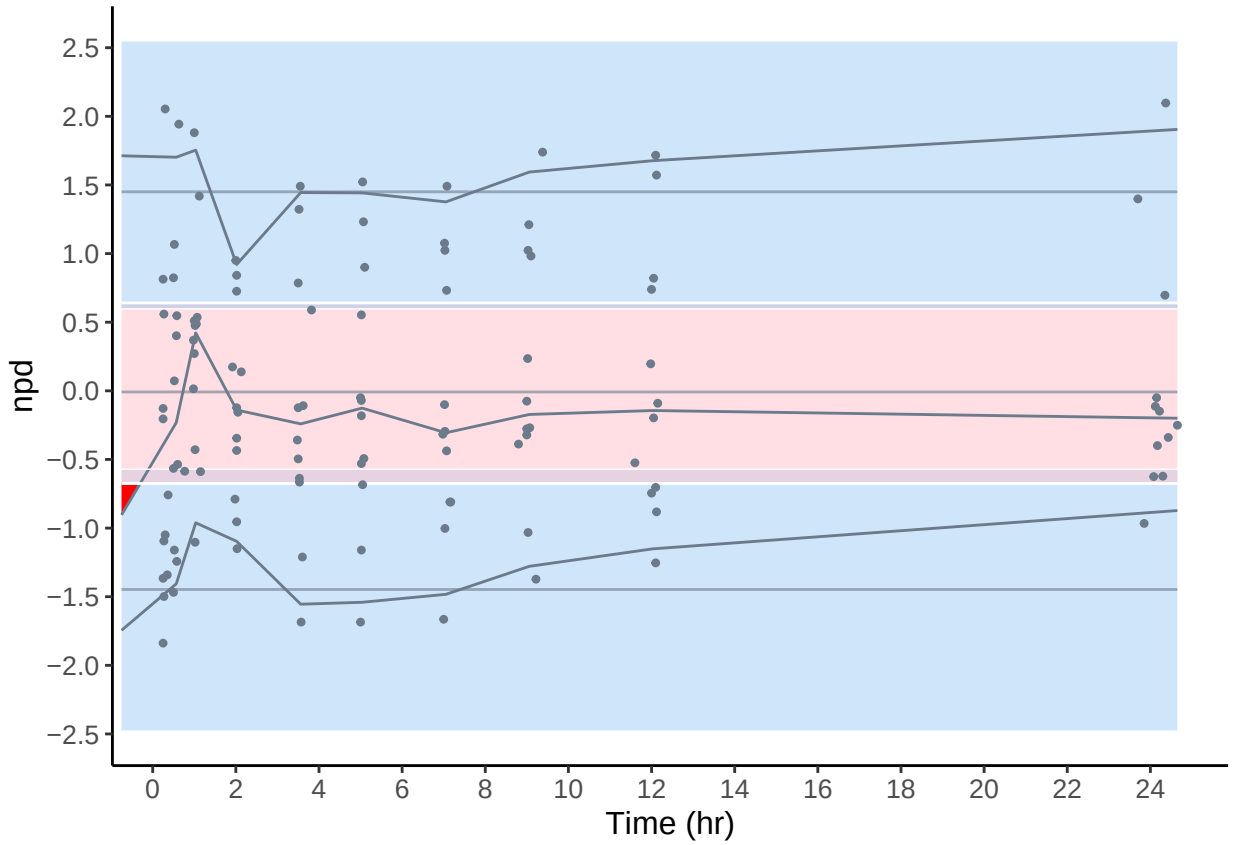



```
# individual npde plots
plot(ynpde, plot.type="x.scatter", covsplit=TRUE, which.cov=c("Weight", "Sex"))
```

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

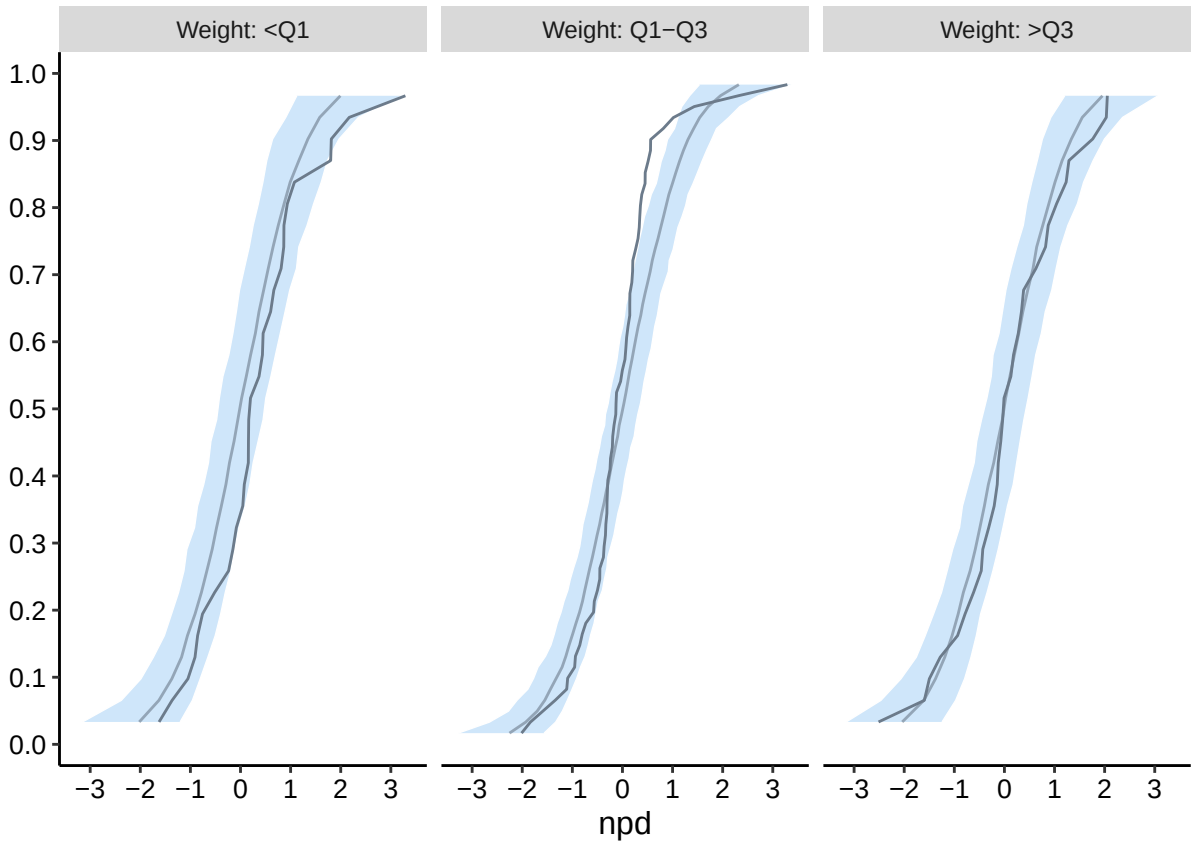

Weight: <Q1
Weight: Q1−Q3
Weight: >Q3
npd
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
0 2 4 6 8 10 12 14 16 18 20 22 24
Time (hr)

```
##
## [[2]]
```

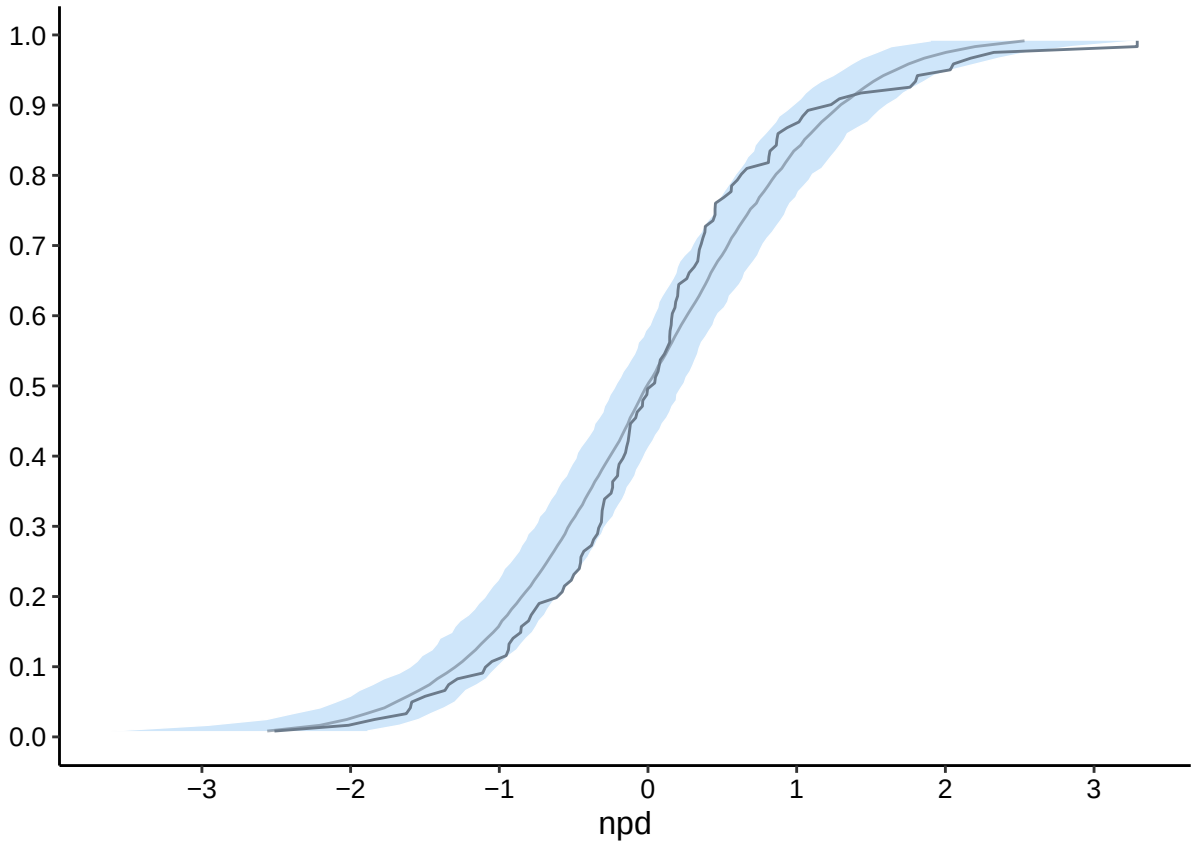



```
plot(ynpde, plot.type="ecdf", covsplit=TRUE, which.cov=c("Weight", "Sex"))
```

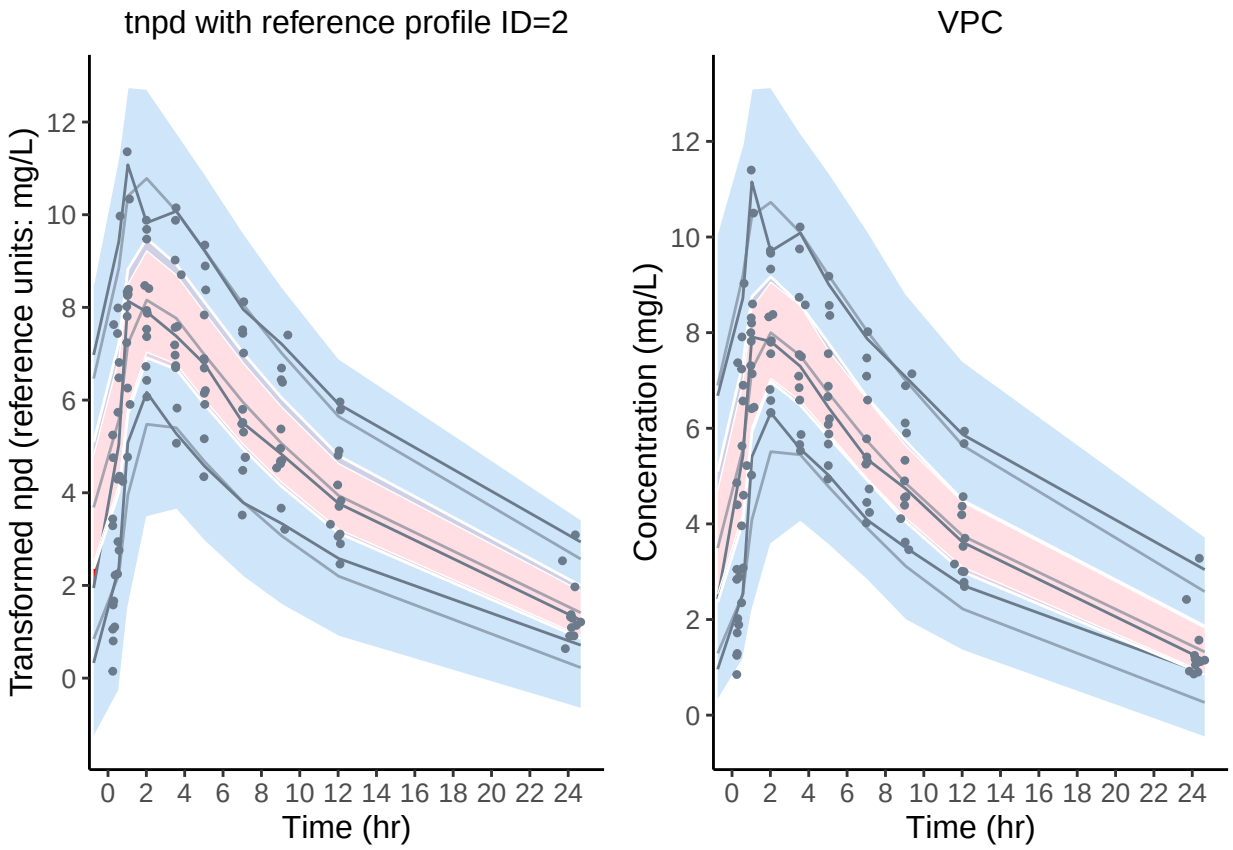
```
## [[1]]
```


Weight: <Q1
Weight: Q1−Q3
Weight: >Q3
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
−3 −2 −1 0 1 2 3
npd

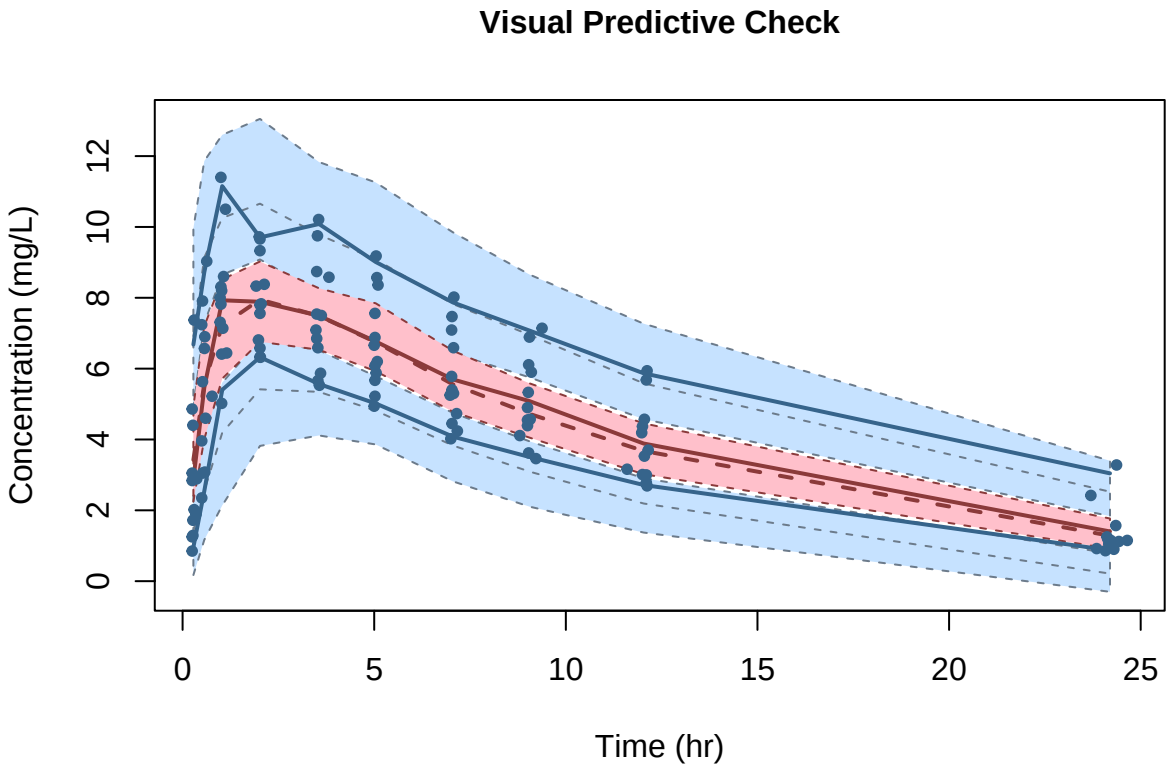
```
##
## [[2]]
```



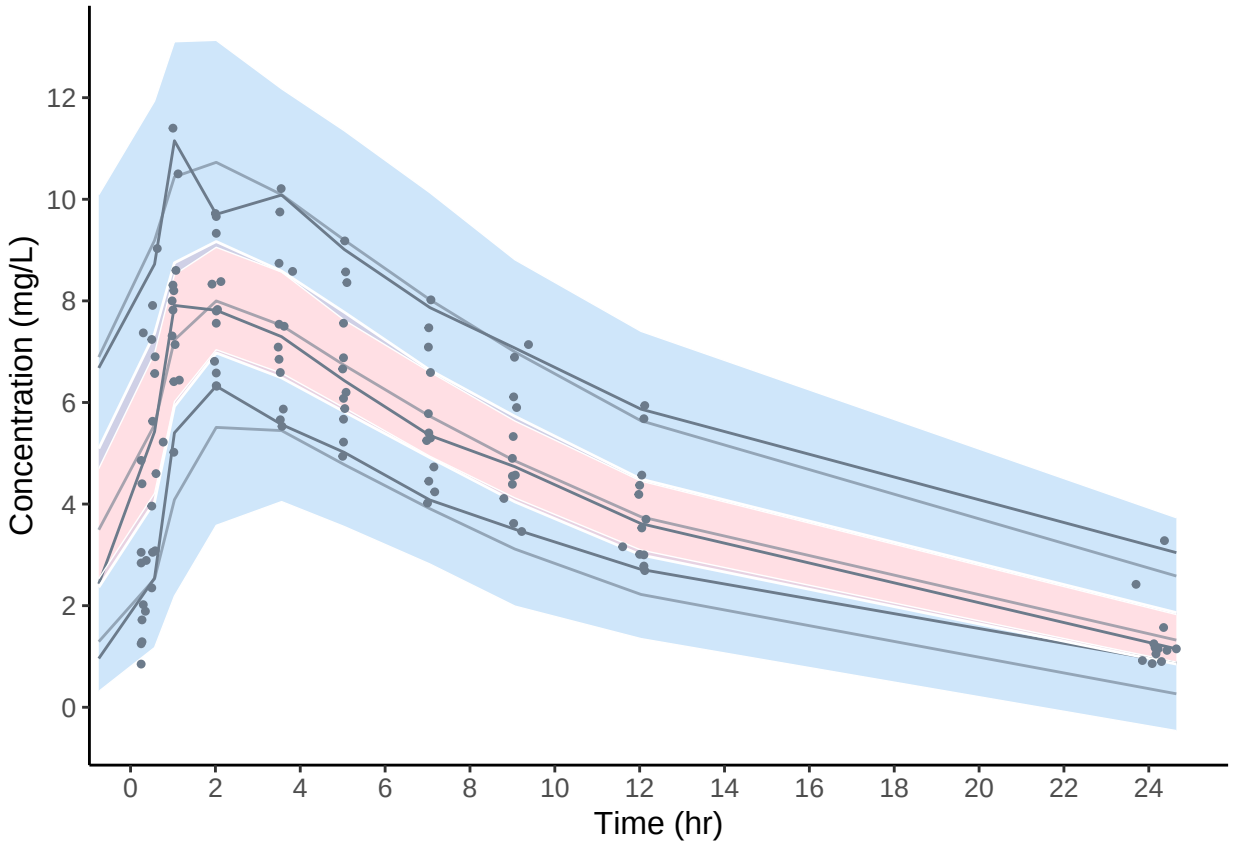

```
plot.tnpde<-plot(ynpde, plot.type="x.scatter", ref.prof=list(Id=2), main="tnpd with reference profile I
plot.vpc<-plot(ynpde, plot.type="vpc", main="VPC")
grid.arrange(grobs=list(plot.tnpde, plot.vpc), nrow=1, ncol=2)
```




```
# VPC
plot(saemix.fit, plot.type="vpc")
```

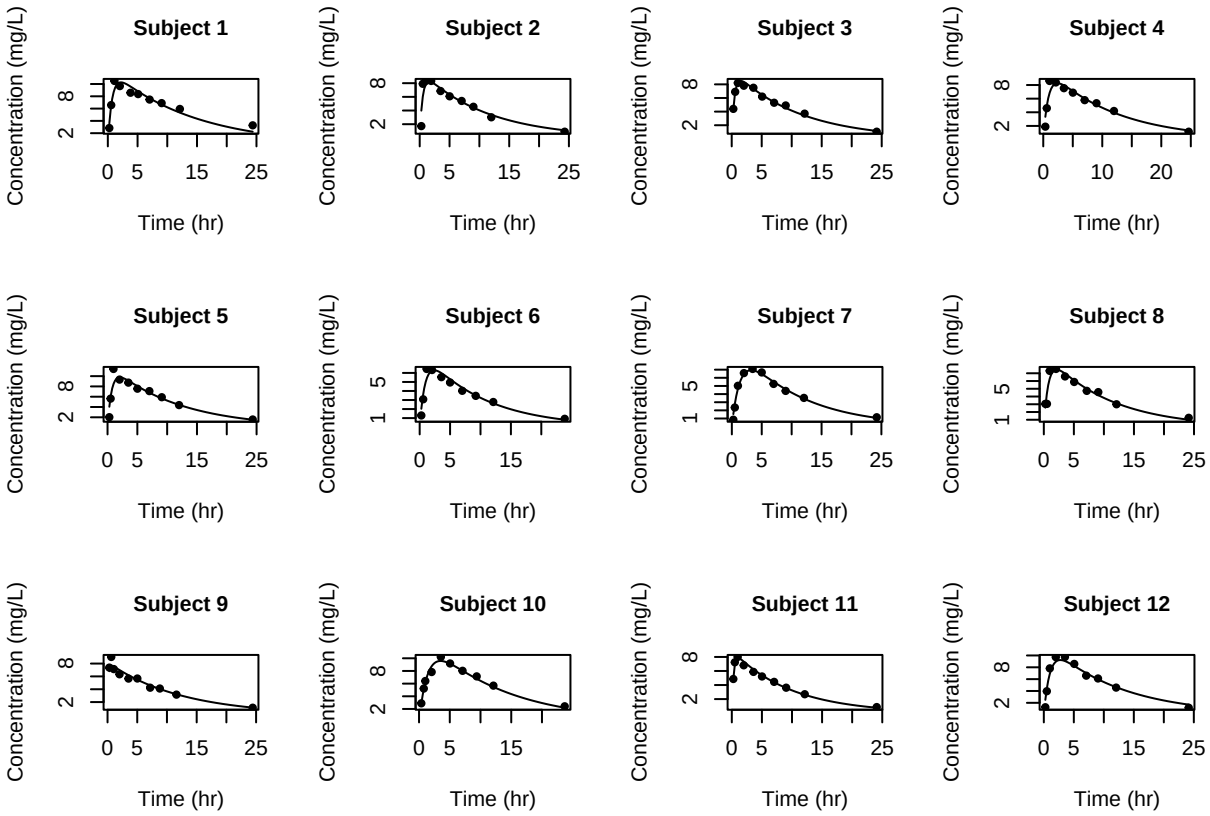


```
plot(ynpde, plot.type="vpc")
```




```
# Individual smoothed plots
plot(saemix.fit, plot.type="individual", smooth=TRUE)
```

```
## Computing WRES and npde ..
```


Subject 1
Subject 2
Subject 3
Subject 4
Subject 5
Subject 6
Subject 7
Subject 8
Subject 9
Subject 10
Subject 11
Subject 12
Concentration (mg/L)
Time (hr)