Case study 2: a one-compartment PK model involving both covariates and BQL

CERTARA

Introduction

- R is one of the most widely used softwares among pharmacometricians to perform data manipulation/visualization and statistical analysis.
- RsNLME provides a R interface to the Phoenix NLME engine to enable users to
  - Define PK/PD models via R objects (package **RsNlme**).
  - Use the "Initial Estimates" shiny app to visually determine a set of reasonable initial values for fixed effects (package **RsNlme**).
  - Perform estimation and simulation in a R environment with the capability of parallelizing the runs using Multicore, MPI and Grids (SGE/Torque/LSF) in-house or hosted on AWS (package **Certara.NLME8**).
  - Access the xpose graphics library for PK/PD models by creating compatible database from NLME results (package **Xpose.Nlme**).

Objectives

Demonstration of RsNLME through a one-compartment PK model involving both covariates and BQL.

- Define the model through **RsNlme**.
- Map model variables to input dataset columns.
- Fit the model, and then use the **xpose.Nlme** package to create commonly used diagnostic plots.
- VPC analysis for the fitted model.

Note: R script and input dataset for this example can be found in C:\Program Files\R\R-n.n.n\library\RsNlme\

Define the model through RsNlme

Structural Model

```
#modelname
ModelName = "OneCpt_IVBolus_ContCovariatesOnClV_BQL_Laplacian"

#definethebasicPKmodel(aone-compartmentmodelwithIVbolus)
model =pkmodel(numCompartments = 1, modelName = ModelName)
```

Covariate Model

$$V = tvV \left(\frac{BW}{30}\right)^{dVdBW} \exp(\eta_V)$$

$$Cl = tvCl \left(\frac{BW}{30}\right)^{dCldBW} \frac{PMA^{Gam}}{PMA^{Gam} + PMA50^{Gam}} \exp(\eta_{Cl})$$

```
#definecovariatesBWandPMA
BW =NlmeCovariateParameter(name = "BW", type = Continuous
                           , continuousType = CovarNumber, centerValue  = "30")
PMA =NlmeCovariateParameter(name = "PMA")

#automaticallyincorporatecovariateBWintothebasicmodel
model =addCovariates(model, covariates = c(BW, PMA)
                           , effects = c("V" = "BW", "Cl" = "BW"))

#mauallyincorporatecovariatePMAintothebasicmodel
structuralParam(model, "Cl") = c(style = Custom
,code= "stparm(Cl=tvCl*(BW/30)^dCldBW*(PMA^Gam/(PMA^Gam+PMA50^Gam))*exp(
 nCl))",extraCode= c("fixef(PMA50=c(, 5, ))", "fixef(Gam=c(, 1, ))"))
```

Residual Error Model

```
#setresidualerrormodel(default:additivemodelwithstandard  #d
eviation=landisBQL=FALSE,wheretheisBQLoptionisusedto  #speci
fywhethertheinputdatasetinvolvesBQLobservationsornot)  resi
dualEffect(model, "C") = c(errorType= Multiplicative
                                  ,SD= "0.1",isBQL= TRUE)
```

Initial Values for Theta and Omega

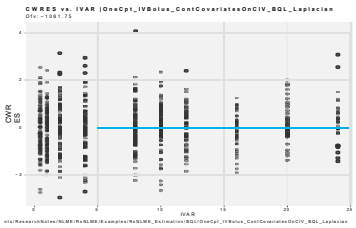
```
#setinitialvaluesforfixedeffects(defaultvaluefortheonerelated
#toacovariateis0;otherwise,itis1)
initFixedEffects(model) = c(tvV = 20, tvCl = 20, dVdBW = 1, dCldBW = 1)

#seti n i t i a l v a l u e sforr a n d o m e f f e c t s(thed e f a u l tvalueis1)  init
RandomEffects(model) = c(Diagonal, isFrozen=FALSE, "nV,nCl", "0.1,0.2")
```

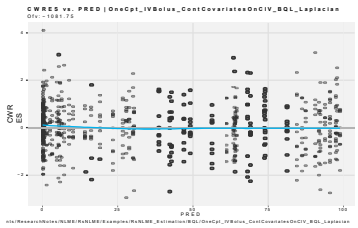
Map model variables to input dataset columns

```
#loadtheinputdataset
dt_InputDataSet = fread("OneCpt_IVBolus_ContCovariatesOnClV_BQL.csv")

#initializemodelmappingandautomaticallymappingsomeofthemodel
#variablestothedatacolumns
initColMapping(model) = dt_InputDataSet

#manuallysetupthemappingfortherestofvariables
modelColumnMapping(model) = c(A1 = "Dose")
```

Fit the model and create diagnostic plots

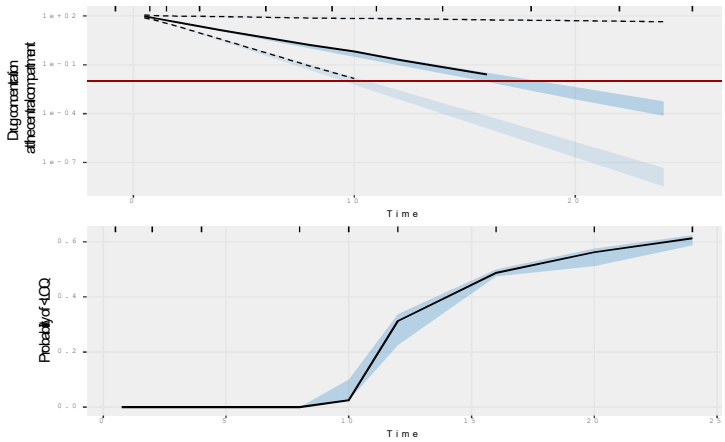
```
#hostsetup:runlocallywithMPIenabled
host =NlmeParallelHost(sharedDirectory = Sys.getenv("NLME_ROOT_DIRECTORY")
                              , parallelMethod = NlmeParallelMethod("LOCAL_MPI")
                              , hostName = "MPI", numCores = 4)
#enginesetup
engineParams =NlmeEngineExtraParams(PARAMS_METHOD = METHOD_LAPLACIAN
                    , PARAMS_NUM_ITERATIONS = 1000, PARAMS_SAND="TRUE")

#fitthemodel
job =fitmodel(host, engineParams, model)

#ImportsresultsofanNLMErunintoxposedatabase
xp =xposeNlme(dir = model@modelInfo@workingDir, modelName = ModelName)

#CreatetheplotfortheCWRESagainsttheindependentvariable
r e s _ v s _ i d v( xp , res = " CWRES " , type = " ps " )
#CreatetheplotfortheCWRESagainstpopulationpredications
res_vs_pred(xp, res = "CWRES", type = "ps")
```





VPC analysis for the fitted model

```
#Acceptheestimatesforfixedeffects,randomeffectsandsigma
modelVPC =acceptAllEffects(model)

modelVPC@dataset@outputFilename= "predout.csv"

#VPCsetup
VPCSetup =NlmeVpcParams(numReplicates = 100, seed = 1)

#RunVPC
job =vpcmodel(host, VPCSetup, m o d e l  =  modelVPC)

#Loadobserveddataandsimulateddata  dt_Obs
Data =getObsData(dt_InputDataSet,
modelDir= modelVPC@modeInfo@workingDir )
dt_SimData =getSimData(input=dt_InputDataSet, simFile="predout.csv",
modelDir= modelVPC@modeInfo@workingDir )

#Usethevpcpackagetocreateavpcplotforun-censoreddatathatshows
#thecensorlimit(LLOQ)asahorizontalline
vpc(sim=dt_SimData, obs=dt_ObsData, lloq=lloq_value, log_y=TRUE, log_y_min=1
 e-9, xlab="Time", ylab="Drug concentration \n at the central compartment")

#Usethevpcpackagetocreateavpcplotfortheprobabilityof
#left-censoreddata
vpc_cens(sim=dt_SimData, obs=dt_ObsData, lloq=lloq_value, xlab="Time")
```



Conclusions

- RsNLME provides R command line access to the Phoenix NLME engine allowing pharmacometricians with little or no knowledge of Phoenix NLME to format and visualize data, build and analyze models, and post-process results.
- RsNLME also provides greater flexibility for advanced Phoenix NLME users to work seamlessly with other R packages within the R environment.