Case study 3: a joint one-compartment PK and indirect response model

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 joint one-compartment PK and indirect response model.

- Define the model through **RsNlme**.
- Map model variables to input dataset columns.
- Fit the model.
- 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

Specify Structural Model

```
#definethePKPDmodel(PK:aone-compartmentmodelwithIVbolus;
#Indirectmodel:inhibitionlimitedontheloss)
model =pkindirectmodel(indirectType =LimitedInhibition, isBuildup =FALSE
          , modelName ="OneCptIVBolus_IndirectInhibLimLoss_FOCE-ELS")
```

Set Structural Model Parameters

```
#SetImax=ilogit(tvlogitImax)withilogitusedtomakesureitis
#between0and1(default:Imax=tvImax*exp(nImax)) structu
ralParam(model, "Imax") =c(style =Logit
          , fixedEffName ="tvlogitImax", hasRandomEffect =FALSE)

#disablerandomeffectforIC50(default:IC50=tvIC 50*exp(nIC50))
structuralParam(model, "IC50") =c(hasRandomEffect =FALSE)
```

Set Initial Values for Theta and Omega

```
#setinitialvaluesforfixedeffects(defaultvaluefortheonerelated
#toacovariateis0;otherwise,itis1)
initFixedEffects(model) =c(tvCl =0.5, tvKin =10, tvKout =0.5)

#setinitialvaluesforrandomeffects(thedefaultvalueis1) in
itRandomEffects(model) =c(Diagonal, isFrozen =FALSE
                  , "nV, nCl, nKin, nKout", "0.01, 0.01, 0.01, 0.01")
```

Set Residual Error Models

```
#settheresidualerrormodelforCObs(default:additivemodelwithSD=1) resid
ualEffect(model, "C") =c(errorType= Multiplicative ,SD= "0.1")

#settheresidualerrormodelforEObs
residualEffect(model, "E") =c(errorType= Multiplicative ,SD= "0.1")
```

Map model variables to input dataset columns

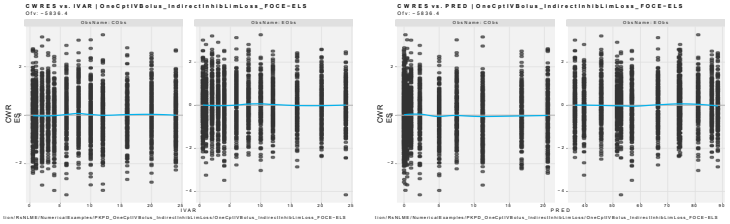
```
#loadtheinputdataset
dt_InputDataSet =fread("OneCptIVBolus_IndirectInhibLimLoss.csv")

#initializemodelmappingandautomaticallymappingsomeofthemodel
  variablestothedatacolumns
initColMapping(model) =dt_InputDataSet

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

Fit the model

```
inputdatasetandmappingfiles NlmeFileNames =NlmeDataset()

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

#runthemodel
job =fitmodel(host, engineParams, model)
```

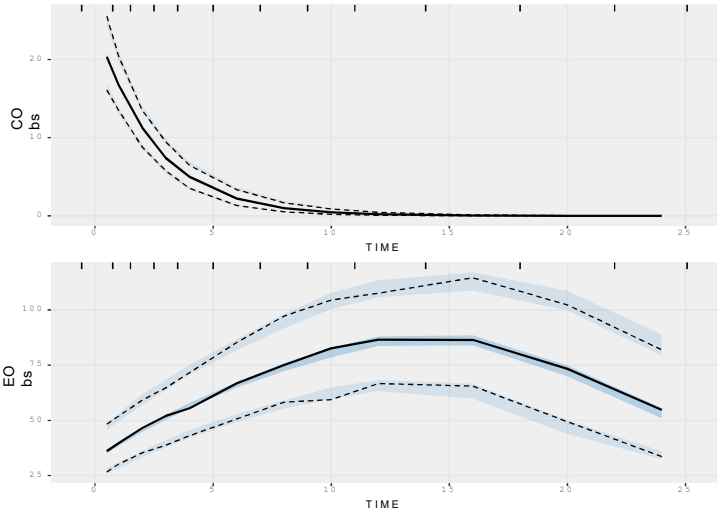
Diagnostic plots

```
#ImportsresultsofanNLMErunintoxposedatabase
xp =xposeNlme(dir ="./",
        modelName ="OneCptIVBolus_IndirectInhibLimLoss_FOCE-ELS")

#CreatetheplotfortheCWRESagainstthein dependentvariable
#foreachobservedvariable
res_vs_idv(xp, res ="CWRES", type ="ps", facets ="ObsName")
#CreatetheplotfortheCWRESagainstp opulatio npred icatio ns
#foreachobservedvariable
res_vs_pred(xp, res ="CWRES", type ="ps", facets ="ObsName")
```



VPC analysis for the fitted model

```
#Acceptthe estimatesforfixedeffects,randomeffectsandsigma
  modelVPC=acceptAllEffects(model)

#Createthedefaultnameforthemodel,inputdatasetandmappingfiles,
#andsettheoutputfilenameto"predout.csv"(defaultname:"out.txt")
NlmeFileNames =NlmeDataset(outputFilename ="predout.csv")

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

#RunVPC
job =vpcmodel(host, VPCSetup, model =modelVPC)

#Loadsimulationinputdataset(thegeneratedpredcheck0.csvputallthe
#observationsinonecolumn)
dt_ObsData =fread("predcheck0.csv")
setnames(dt_ObsData, c("IVAR", "ID5"), c("TIME", "ID"))
dt_ObsData_CObs =dt_ObsData[ObsName =="CObs"] dt_O
bsData_EObs =dt_ObsData[ObsName =="EObs"]

#loadsimulateddata
dt_SimData =fread("predout.csv")
setnames(dt_SimData, c("ID5", "IVAR"), c("ID", "TIME"))
dt_SimData_CObs =dt_SimData[OBSNAME=="CObs"] dt_S
imData_EObs =dt_SimData[OBSNAME=="EObs"]

#UsethevpcpackagetocreateaVPCplotforCObs
vpc(sim =dt_SimData_CObs, obs =dt_ObsData_CObs, ylab ="CObs")

#UsethevpcpackagetocreateaVPCplotforEObs
vpc(sim =dt_SimData_EObs, obs =dt_ObsData_EObs, ylab ="EObs")
```



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.