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

```
# define the PKPD model (PK: a one-compartment model with IV bolus;
# Indirect model: inhibition limited on the loss)
model = pkindirectmodel(indirectType = LimitedInhibition, isBuildup = FALSE
          , modelName = "OneCptIVBolus_IndirectInhibLimLoss_FOCE-ELS")
```

Set Structural Model Parameters

```
# Set Imax = ilogit(tvlogitImax) with ilogit used to make sure it is
# between 0 and 1 (default: Imax = tvImax * exp(nImax))
structuralParam(model, "Imax") = c(style = Logit
         , fixedEffName = "tvlogitImax", hasRandomEffect = FALSE)

# disable random effect for IC50 (default: IC50 = tvIC50 * exp(nIC50))
structuralParam(model, "IC50") = c(hasRandomEffect = FALSE)
```

Set Initial Values for Theta and Omega

```
# set initial values for fixed effects (default value for the one related
# to a covariate is 0; otherwise, it is 1)
initFixedEffects(model) = c(tvCl = 0.5, tvKin = 10, tvKout = 0.5)

# set initial values for random effects (the default value is 1)
initRandomEffects(model) = c(Diagonal, isFrozen = FALSE
                     , "nV, nCl, nKin, nKout", "0.01, 0.01, 0.01, 0.01")
```

Set Residual Error Models

```
# set the residual error model for CObs (default: additive model with SD=1)
residualEffect(model, "C") = c(errorType = Multiplicative, SD = "0.1")

# set the residual error model for EObs
residualEffect(model, "E") = c(errorType = Multiplicative, SD = "0.1")
```

Map model variables to input dataset columns

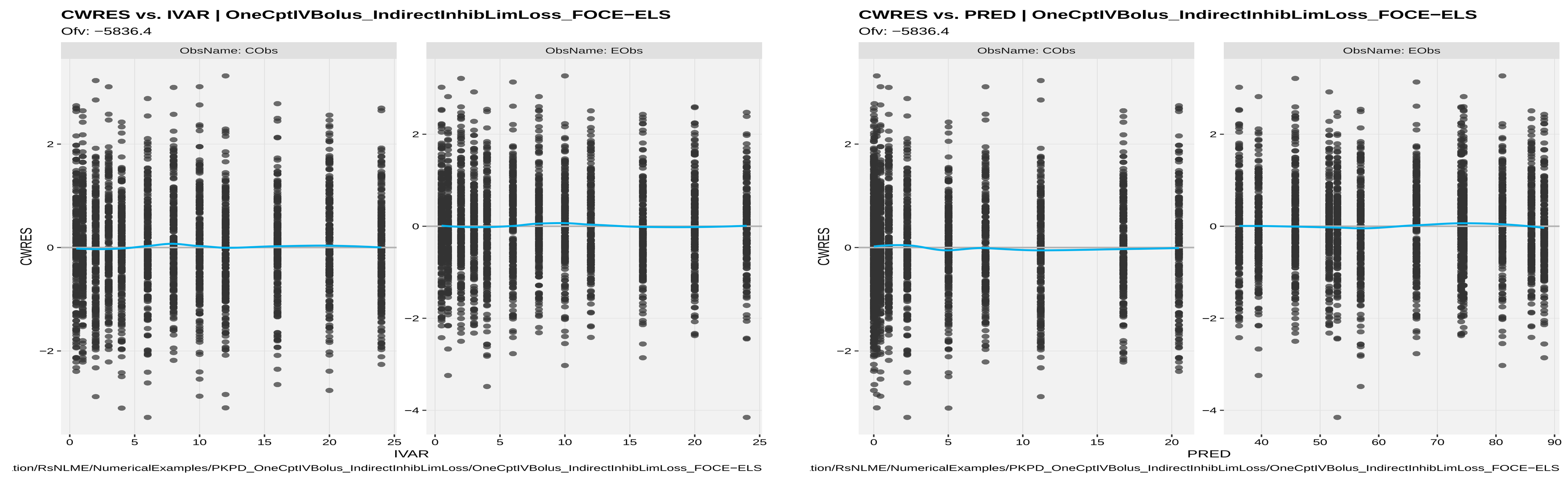
```
# load the input dataset
dt_InputDataSet = fread("OneCptIVBolus_IndirectInhibLimLoss.csv")

# initialize model mapping and automatically mapping some of the model
  variables to the data columns
initColMapping(model) = dt_InputDataSet

# manually set up the mapping for the rest of variables
modelColumnMapping(model) = c(A1 = "Dose")
```

Fit the model

```
# create the default name for the model, input dataset and mapping files
NlmeFileNames = NlmeDataset()

# host setup: run locally with MPI enabled
host = NlmeParallelHost(sharedDirectory = Sys.getenv("NLME_ROOT_DIRECTORY")
                        , parallelMethod = NlmeParallelMethod("LOCAL_MPI")
                        , hostName = "MPI", numCores = 4)
# engine setup
engineParams = NlmeEngineExtraParams(PARAMS_METHOD = METHOD_FOCE_ELS
                   , PARAMS_NUM_ITERATIONS = 1000, PARAMS_SAND="TRUE")

# run the model
job = fitmodel(host, NlmeFileNames, engineParams, model)
```

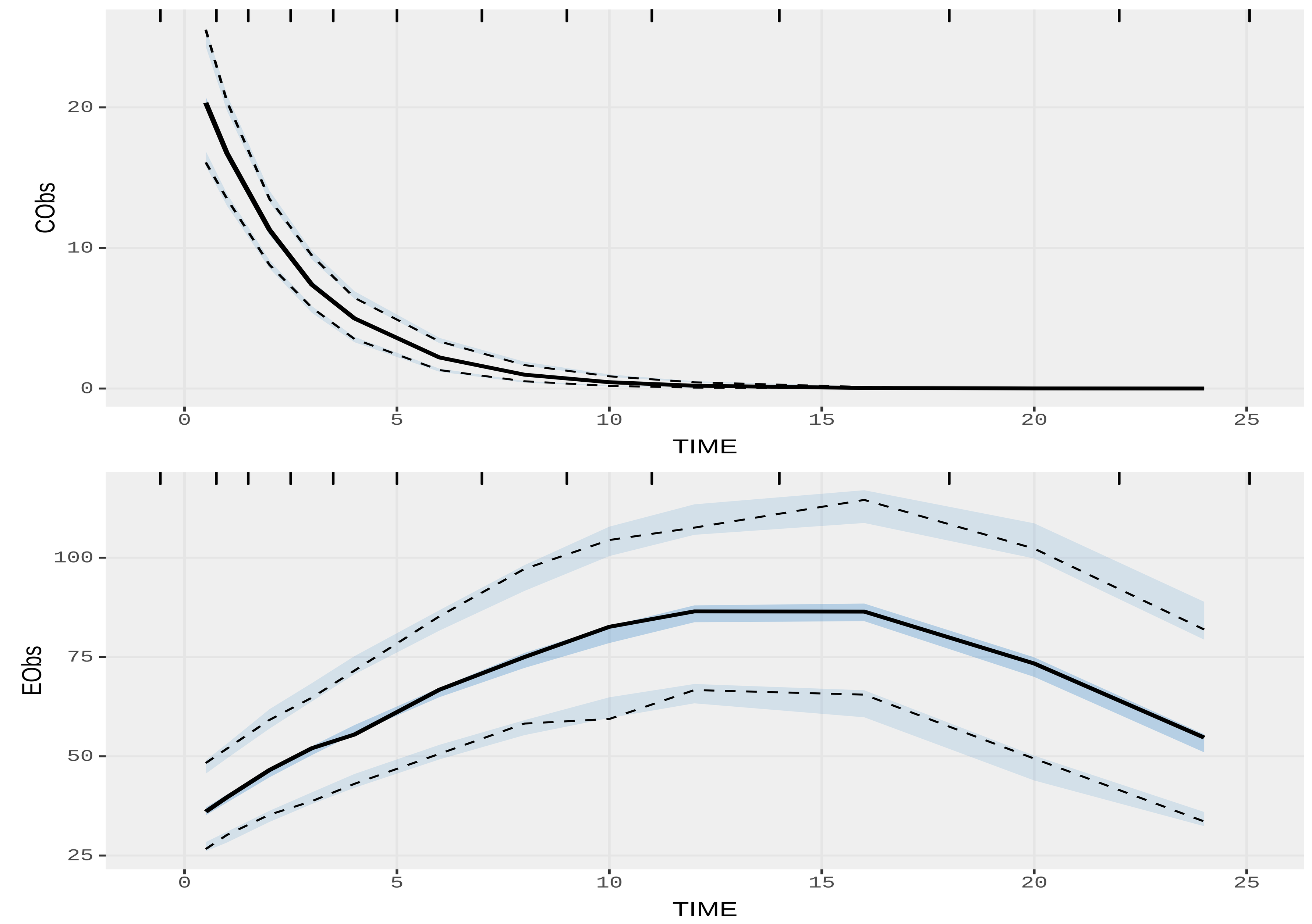
Diagnostic plots

```
# Imports results of an NLME run into xpose database
xp = xposeNlme(dir = "./",
       modelName = "OneCptIVBolus_IndirectInhibLimLoss_FOCE-ELS")

# Create the plot for the CWRES against the independent variable
# for each observed variable
res_vs_idv(xp, res = "CWRES", type = "ps", facets = "ObsName")
# Create the plot for the CWRES against population predications
# for each observed variable
res_vs_pred(xp, res = "CWRES", type = "ps", facets = "ObsName")
```



VPC analysis for the fitted model

```
# Accept the estimates for fixed effects, random effects and sigma
modelVPC = acceptAllEffects(model)

# Create the default name for the model, input dataset and mapping files,
# and set the output file name to "predout.csv" (default name: "out.txt")
NlmeFileNames = NlmeDataset(outputFilename = "predout.csv")

# VPC setup
VPCSetup = NlmeVpcParams(numReplicates = 100, seed = 1)

# Run VPC
job = vpcmodel(host, NlmeFileNames, VPCSetup, modelVPC)

# Load simulation input dataset (the generated predcheck0.csv put all the
# observations in one column)
dt_ObsData = fread("predcheck0.csv")
setnames(dt_ObsData, c("IVAR", "ID5"), c("TIME", "ID"))
dt_ObsData_CObs = dt_ObsData[ObsName == "CObs"]
dt_ObsData_EObs = dt_ObsData[ObsName == "EObs"]

# load simulated data
dt_SimData = fread("predout.csv")
setnames(dt_SimData, c("ID5", "IVAR"), c("ID", "TIME"))
dt_SimData_CObs = dt_SimData[OBSNAME == "CObs"]
dt_SimData_EObs = dt_SimData[OBSNAME == "EObs"]

# Use the vpc package to create a VPC plot for CObs
vpc(sim = dt_SimData_CObs, obs = dt_ObsData_CObs, ylab = "CObs")

# Use the vpc package to create a VPC plot for EObs
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.