

BioUML: plugin for population-based modeling

Kiselev I.N.^{1,2}, Kolpakov F.A.^{1,2}

¹*Institute of Systems Biology, Ltd*

²*Design Technological Institute of Digital Techniques SB RAS*

Abstract

Motivation and Aim: Non-linear mixed effects (NLME) model is an efficient tool for analyzing of population data and estimating of model parameters. It is widely used in pharmacological modeling, but may be applied to a variety of fields. NLME model contains core structural model which is the mathematical model of studying process (e.g. drug dynamics in organism) and suggests distribution of model parameters across population. nlme is a library written in R language implementing log-likelihood approach for NLME models creation and analysis. Core structural model should be supplied as R function which complicates usage of sophisticated mathematical models. Library nlmeODE allows usage of ODEs inside nlme, but it has certain restrictions on possible models and textual format complicates using of complex ODE systems.

Results: We have implemented plugin for BioUML platform for population-based modeling. Plugin includes graphical notation facilitating work with detailed and large-scaled models, parameter estimation is performed by R function, supplied (using rJava) with Java simulator from BioUML. It grants possibility to use any model created in BioUML as a structural model. Executing R script is automatically generated on the basis of the diagram.

Availability: Plugin is freely available as a part of BioUML software at www.biouml.org.

Introduction

Non-linear mixed effects modeling is an approach for studying of population-based data and using it for model fitting. Core of NLME model is a structural model which describes modeled system.

NLME model may be defined as follows:

$$\begin{aligned}y_{ij} &= f(\phi_{ij}, v_{ij}) + \varepsilon_{ij}, \quad \varepsilon_{ij} \sim N(0, \sigma^2) \\ \phi_{ij} &= A_{ij}\beta + B_{ij}b_i, \quad b_i \sim N(0, \Psi) \\ i &= 1, \dots, n, \quad j = 1, \dots, m\end{aligned}$$

Where:

y – output (observable) variable,

f – structural model,

ϕ – vector of structural model parameters,

v – regression variable (e.g. time),

ε - residual error, ε_{ij} are independent and equally distributed,

A, B – design matrices,

β - fixed effect,

b - random effect, b_i are independent and equally distributed.

Estimation task implies that we have observations for y and our task is to estimate values of $\beta, b_i, \Psi, \phi, \sigma^2$ and ε_{ij} .

Simulation task implies simply simulation of the structural model i.e. parameters ϕ values are known and we should obtain y .

This approach is widely used in pharmacological models, where structural model is usually so called PK/PD (Pharmacokinetic/Pharmacodynamic) model which describes drug concentration change in organism compartments and its effect on organism. NLME approach allows using of PK/PD model not for single patient but for population of patients.

There are a number of software products (mostly commercial) providing tools for creation and work with population-based data and NLME models. Most known products are NONMEM (<http://www.lixoft.eu/>) and MONOLIX [1]. WinBUGS/PKBUGS [2] is a free tool utilizing Bayesian approach. There are also libraries for R language such as nlme [3], nlmeODE [4], seamix.

In past year, new language for NLME models description, PharmML [5] has appeared. It aims to become a standard for models description and exchange between different tools.

BioUML

BioUML is an open source integrated Java platform for modeling of biological systems. It provides access to different databases, tools for mathematical modeling, statistical analysis and numerical calculations including Ordinary Differential Equations (ODE) with discrete events, stochastic, composite and agent-based models. BioUML utilizes visual modeling approach which implies:

1. Creation and editing of the model as visual diagram.
2. Automatic Java code generation on the basis of diagram. Generated code is used for numerical simulations.

BioUML supports Systems Biology Markup Language (SBML, [6]) for models formal description and Systems Biology Graphical Notation (SBGN, [7]) for visual representation. It also has its own graphical notation.

Plugin-based structure of the platform facilitates its extending by new software plugins.

Our aim is to provide BioUML with tools for NLME modeling. For this purpose we have chosen nlme library in R language. The reason is that BioUML supports execution of R and java-script commands in embedded console.

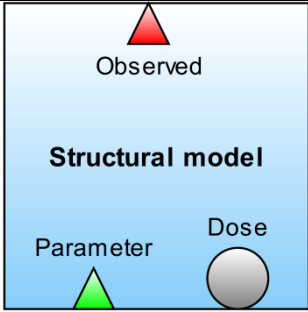
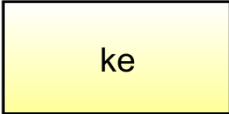
Graphical notation

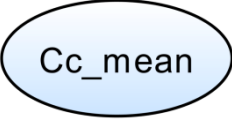
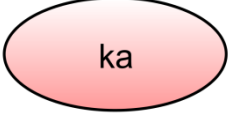
We have implemented special diagram type for NLME models representation. BioUML interface for NLME models is represented on [Figure 1]. Diagram contains elements referring to another diagram as a structural model. Particularly it may refer to diagram representing:

1. ODE model with BioUML/SBGN notation
2. SBML model imported in BioUML (BioUML/SBGN notation)
3. Composite model created in BioUML – model which contains other models as parts. Simulation is performed utilizing either formalism transformation (in the case of ODE formalism for all submodels) or agent-based approach in the case of different formalisms.
4. Stochastic model.

Parameter distribution is defined by variable elements in the WinBUGS-like way (where we have stochastic, function and constant elements connected with each other). For detailed information about all NLME diagram elements and their graphical notation see [Table 1]. Parameter properties such as type, distribution, transformation (e.g. log, logit, none), initial value, comment, may be specified in “Population variables” tab (see [Figure 2])

Table 1. NLME diagram graphical notation.

Graphical notation	Description															
	<p>Structural model. Refers another diagram in BioUML. It may contain port elements. Each port corresponds to structural model variables. There three port types:</p> <ol style="list-style-type: none"> 1. Parameter – variable that should be estimated. 2. Observed – variable that is a result of model simulation 3. Other – variable that should not be estimated, but is connected with table data somehow (e.g. dose value) <p>Ports may be used for connection with variables on NLME diagram which defines distributions and dependencies on covariates.</p>															
<p>conc ~ Time Subject</p> <table border="1" data-bbox="284 1171 651 1368"> <thead> <tr> <th>Subject</th> <th>Wt</th> <th>Dose</th> <th>Time</th> <th>conc</th> </tr> </thead> <tbody> <tr> <td>grouping</td> <td>weight</td> <td>dose</td> <td>time</td> <td>Cc</td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table>	Subject	Wt	Dose	Time	conc	grouping	weight	dose	time	Cc						<p>Table element. Refers to table data collection in BioUML. Target table may contain information about dosing regimen and experimental data. It also establishes mapping between table columns (first row on element image) and variables on NLME diagram (second row). For given element table column “Wt” corresponds to model variable “weight”, column “Dose” to variable “dose” and so on. Header contains formula which indicates dependency between table columns. In that case formula is “conc ~ Time Subject” which means that conc is dependent on time and is grouped according to data in “Subject” column.</p>
Subject	Wt	Dose	Time	conc												
grouping	weight	dose	time	Cc												
	<p>Constant parameter. Variable that does not depend on other variables except maybe time. E.g. parameter with only fixed effects and no covariate dependencies or dosing variable.</p>															

	<p>Function. Variable which value is a (deterministic) function of other variables.</p>
	<p>Random variable. Defines distribution and may depend on other variables which then are used as parameters for this distribution.</p>

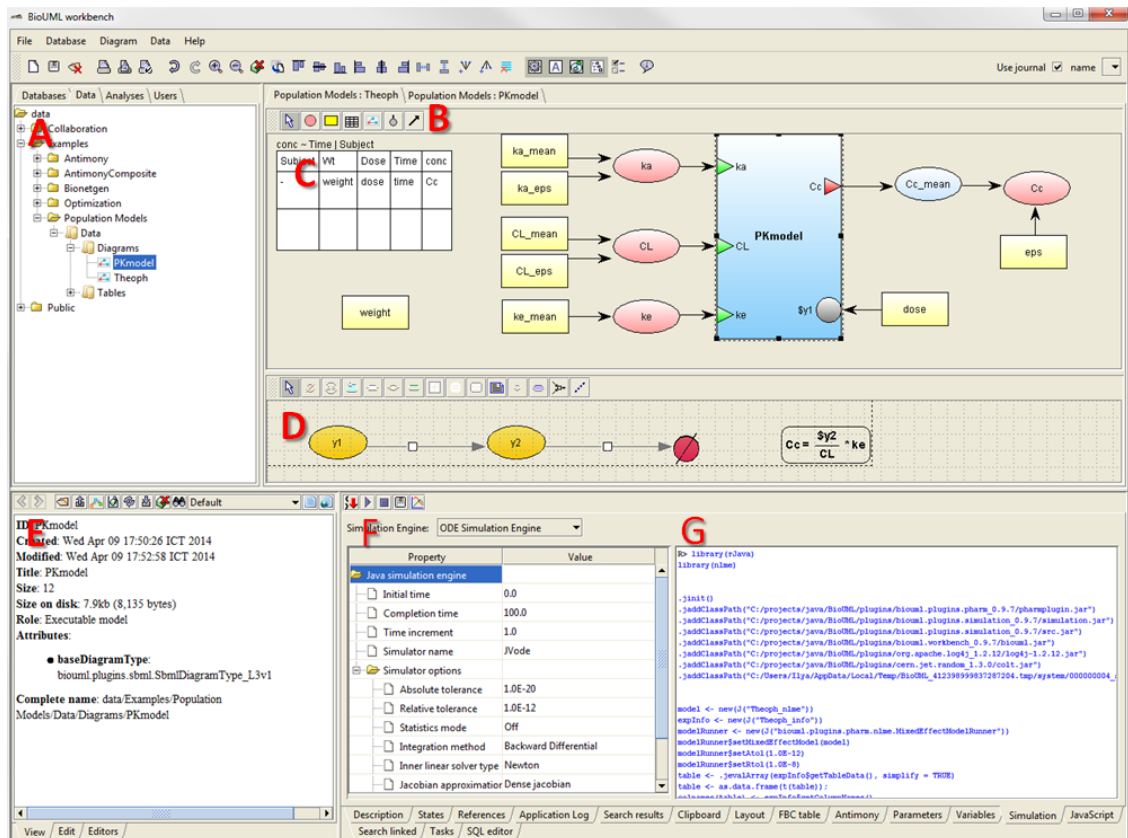


Figure 1. BioUML interface for NLME model creation. A – model repository, B – toolbar for selected model, C – diagram for selected model (NLME), D – panel for structural model, E – description for selected element, F – simulation/estimation options, G – R console with generated script.

	Name	Initial value	Type	Distribution	Transform...	Comment
0	CL	-3.2	stochastic	Normal	log	Clearance
1	Cc	0.0	stochastic	Normal	-	Concentration
2	Cc_mean	0.0	function	-	-	
3	dose	0.0	constant	-	-	
4	eps	0.0	stochastic	Normal	-	Residual error
5	ka	-2.5	stochastic	Normal	log	Transfer rate
6	ke	0.5	constant	-	log	Elimination rate

Figure 2. Panel “Population variables”

Test example

As an example of NLME model created in BioUML we will use simple NLME model containing 1-compartmental PK model as a structural model

Structural model:

$$\begin{cases} \frac{dy_1}{dt} = -ka * y_1 \\ \frac{dy_2}{dt} = ka * y_1 - ke * y_2 \\ Cc = \frac{y_2 * ke}{CL} \end{cases}$$

Although in this simple case we may solve this system analytically, generally to obtain result of structural model, we should numerically simulate ODE system. Model visual diagram created in BioUML is shown on [Figure 3].

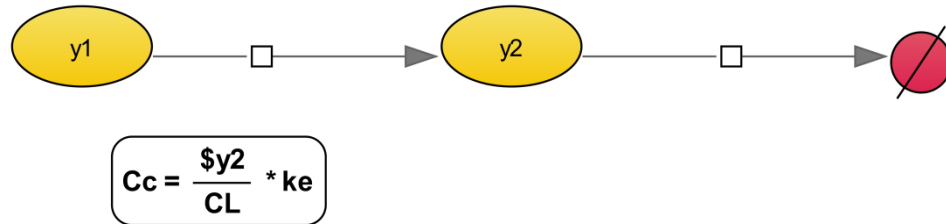


Figure 3. Simple PK model in BioUML (using SBGN)

Observation model:

$$Cc = Cc_{mean} + \epsilon$$

Parameters model:

$$\begin{cases} ka = ka_{mean} + ke_{\epsilon} \\ Cl = Cl_{mean} + Cl_{\epsilon} \\ ke = ke_{mean} \end{cases}$$

Dosing regimen:

$$\text{if } (time == doseTime) \text{ do: } y_1 = doseValue$$

It means that when model time reaches “doseTime”, we should assign y_1 with “doseValue”. Dose time and value may be different among subjects. Note that this rule is not explicitly depicted on PK model – because it is the part of NLME model. Corresponding event is adding into Structural model during preprocessing before Java code is generated.

Visual diagram of the described NLME model is depicted on [Figure 4].

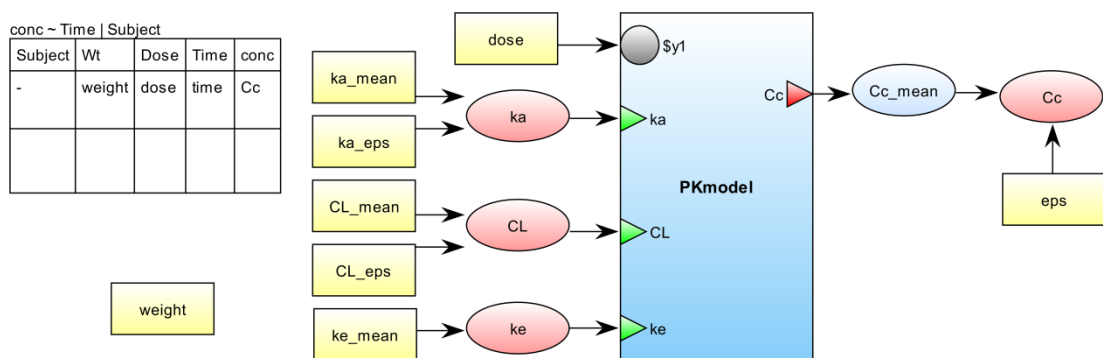


Figure 4. NLME model in BioUML

Parameter estimation

As a tool for NLME model estimation and analysis we picked nlme []. It requires structural model in the form of R function. Passing simple non-linear function is easy, however when structural model gets complicated it demands significant effort to use nlme. Library nlmeODE [2] is designed for using ODE systems. User defines equations in textual forms, then R function for simulation is automatically generated. Library has

certain limitations and textual form makes creation of complex large-scaled models complicated. We use the same approach – generating R function which calls Java classes generated by BioUML. It makes possible using of all model formats and numerical simulators embedded in BioUML for structural model simulation. On the other hand, visual approach facilitates complex model creation and usage. Overall scheme is represented on [Figure 5].

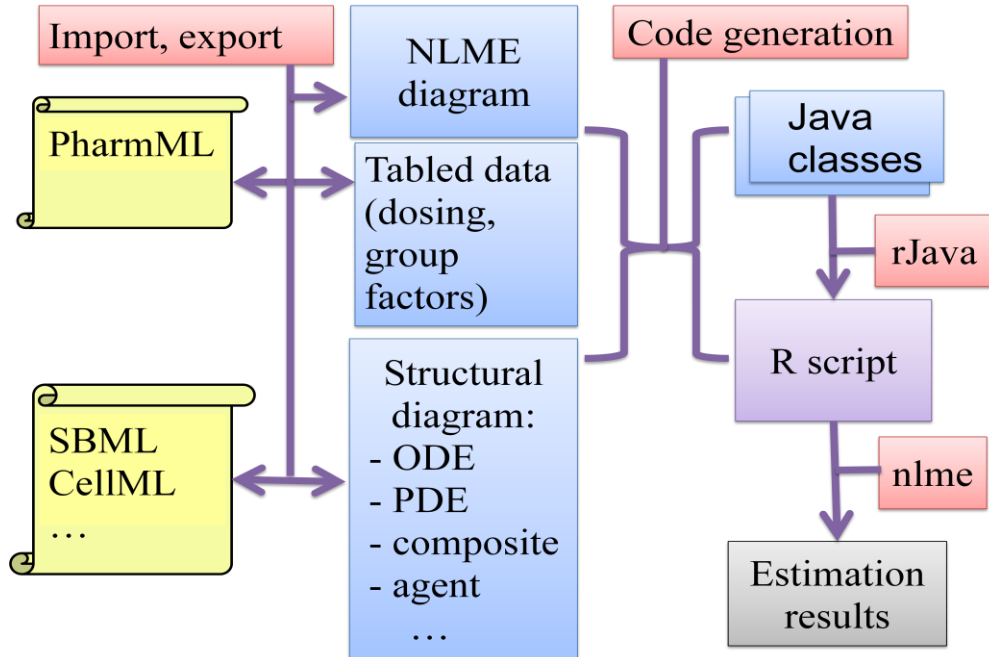


Figure 5. Overall plugin scheme.

Estimation process goes as follows:

1. Java class for numerical model is generated on the basis of structural model
2. Java classes representing NLME model and experiment info form table data are generated.
3. R script is generated. R script uses “rJava” library for access to generated Java classes and “nlme” library for parameters estimation procedure. Generated Java classes are wrapped by R function which takes as arguments parameters value, conduct numerical simulations and returns results for observed variable. This function is passed to “nlme”.
4. R script is evaluated in BioUML R console. Result is also outputted into console, all graphics generated by R are also demonstrated to user (see [Figure 6] for example).
5. After model is created, user may continue updating it and analyzing through R console or by modifying visual diagram and rerunning estimation process.

Thus we combine BioUML visual modeling and numerical simulation with power of R. Generated script for Theopheline drug dynamics with wimple PK model:

```
library(rJava)
library(nlme)
.jinit()
```

```
#initializing necessary libraries for BioUML simulation engine usage
.jaddClassPath("C:/projects/java/BioUML/plugins/biouml.plugins.pharm_0.9.6/pharmpl
ugin.jar")
.jaddClassPath("C:/projects/java/BioUML/plugins/biouml.plugins.simulation_0.9.6/sim
ulation.jar")
```

```
.jaddClassPath("C:/projects/java/BioUML/plugins/biouml.plugins.simulation_0.9.6/src.jar")
.jaddClassPath("C:/projects/java/BioUML/plugins/biouml.workbench_0.9.6/biouml.jar")
.jaddClassPath("C:/projects/java/BioUML/plugins/org.apache.log4j_1.2.12/log4j-1.2.12.jar")
.jaddClassPath("C:/projects/java/BioUML/plugins/cern.jet.random_1.3.0/colt.jar")
.jaddClassPath("C:/Users/Ilya/AppData/Local/Temp/BioUML_8642193711767795961.tmp/system/000000004_simulation")
```

```
#generated Java class instantiating
model <- new(J("Theoph_nlme"))
expInfo <- new(J("Theoph_info"))
modelRunner <- new(J("biouml.plugins.pharm.nlme.MixedEffectModelRunner"))
modelRunner$setMixedEffectModel(model)
modelRunner$setAtol(1.0E-12)
modelRunner$setRtol(1.0E-8)
```

```
#experimental data initializing
table <- .jevalArray(expInfo$getTableData(), simplify = TRUE)
table <- as.data.frame(t(table));
colnames(table) <- expInfo$getColumnNames()
Data <- groupedData( conc ~ Time | Subject,
data = table,
labels = list( x = "Time", y = "Concentration"))
```

```
#wrapper function which performs numerical simulations
f = function( CL,ka,ke, time, subject)
{
  parametersMap <- new(J("java.util.HashMap"))
  parametersMap$put("CL", CL)
  parametersMap$put("ka", ka)
  parametersMap$put("ke", ke)
  result = modelRunner$calculate(parametersMap, .jarray(time), .jarray(subject))
  return (result)
}
```

```
#nlme call: estimation procedure starts here
Theoph.nlme <- nlme(conc ~ f( CL,ka,ke, Time, Subject),
  data = Data, fixed=CL+ka+ke~1, random = pdDiag(CL+ka~1),
  start=c(CL = -3.2,ka = -2.5,ke = 0.5),
  method = "ML",
  control=list(returnObject=TRUE,msVerbose=TRUE),
  verbose=TRUE)
```

```
#results plotting
p <- plot(augPred(Theoph.nlme,level=0:1))
```

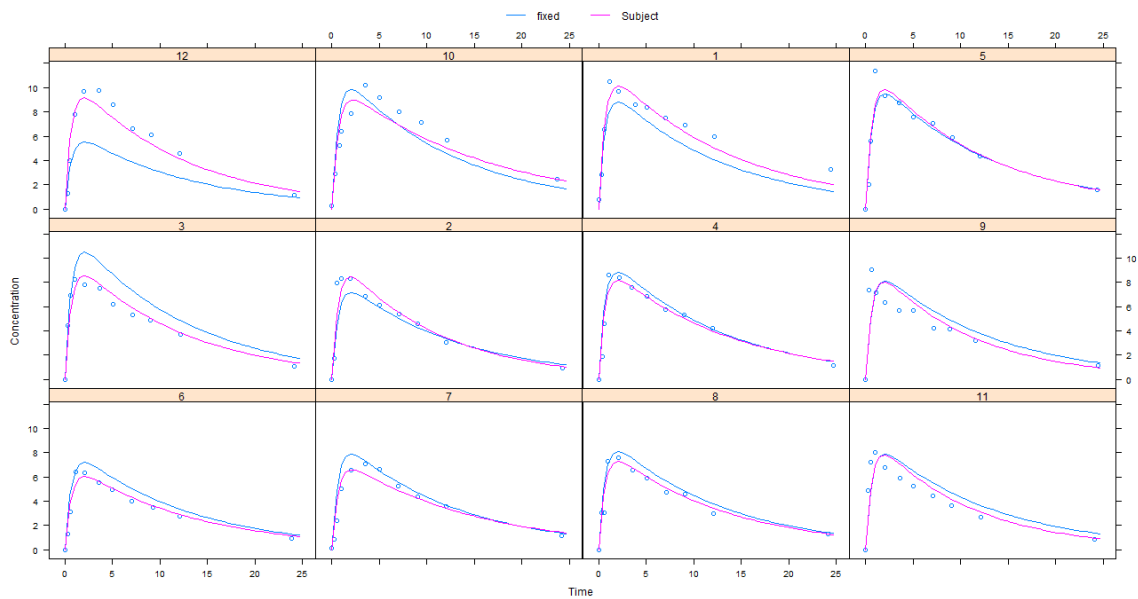


Figure 6. Results of Theophiline drug concentration in BioUML using R.

Conclusions

We have created plugin for BioUML platform providing possibility for visual creation, simulation and parameter estimation of NLME models. As an engine for estimation we used “nlme” library for R language. Structural model is created visually as a diagram then Java-code is generated and passed to R using “rJava” library. Comparison of BioUML with results of nlme with nlmeODE combination are shown in [Table 2]. As one may see, BioUML is nearly 5 times faster than nlmeODE. However these results are preliminary and should not be considered as reliable.

This may be explained by the fact that BioUML uses code generation instead of model interpretation, which is less time-consuming especially if simulation is performed many times during one estimation.

Developed software combines visual representation of mixed-effects models, modeling and simulation tools in BioUML for structural model simulation, and R algorithms for work with mixed-effects models.

Plugin is available as a part of BioUML at www.biouml.org.

Table 2. Comparison: BioUML vs nlmeODE. Both are used in nlme library as numerical simulators for structural model.

	BioUML	nlmeODE
<i>Duration (s)</i>	23.94	108.84
<i>Loglike</i>	-181.177	-177.038
<i>AIC</i>	374.354	366.076
<i>BIC</i>	391.6508	383.3728
<i>Fixed effects</i>	<i>ka</i> = 0.468757 <i>ke</i> = -2.458947 <i>CL</i> = -3.228863	<i>ka</i> = 0.4672482 <i>ke</i> = -2.4557651 <i>CL</i> = -3.227748
<i>Random effects</i>	<i>ka</i> = 0.6343257 <i>CL</i> = 0.217	<i>ka</i> = 0.6433165 <i>CL</i> = 0.1665241
<i>Residual error</i>	0.7102204	0.709441

References

1. Beal, S., Sheiner, L.B., Boeckmann, A., & Bauer, R.J., NONMEM User's Guides. (1989-2009), Icon Development Solutions, Ellicott City, MD, USA, 2009.
2. Lunn, D.; Spiegelhalter, D.; Thomas, A.; Best, N. (2009). "The BUGS project: Evolution, critique and future directions". *Statistics in Medicine* **28** (25): 3049–3067.
3. Moodie, S.L., et al. PharmML: The Pharmacometrics Markup Language. Language Specification. 2013. Available at www.pharmml.org.
4. Pinheiro J.C., Bates D.M. Mixed-Effect Models in S and S-PLUS, Springer-Verlag, New York, NY, 2000. DOI: 10.1007/b98882.
5. Tornøe C.W., et al. Non-linear mixed-effects pharmacokinetic/pharmacodynamic modelling in NLME using differential equations. *Computer Methods and Programs in Biomedicine*, 2004. 76(1) 31-40. DOI: 10.1016/j.cmpb.2004.01.001.
6. Hucka, M. et al. The Systems Biology Markup Language (SBML): A medium for representation and exchange of biochemical network models. *Bioinformatics*, 2003, vol. 19, no. 4, pp. 524–531. DOI: 10.1093/bioinformatics/btg015.
7. Nicolas Le Novère et al. The Systems Biology Graphical Notation. *Nature Biotechnology*, 2009, 27, 735 – 741. DOI: 10.1038/nbt.1558.