



MONOLIX
PROJECT - MLXTRAN

Version 4.3.1

MARCH 2014

A software for the analysis of nonlinear mixed effects models

Maximum likelihood estimation

Model selection

Hypothesis testing

Graphical analysis

Data simulation

• • •

				I	M	P	S T O C H S A S T I C E M	R	T	A	N	M C M C	E	S	A	H M M	P	L	I	N	G
S	I	M	U	L	A			E	D	A	N	N	S A E M	A	L	I	N	G			
					M			T	R	O	P	O	L	I	S						

MONOLIX[®](MOdèles NON LInéaires à effets miXtes) is a platform of reference for model-based drug development. It combines the most advanced algorithms with unique ease of use.

Pharmacometricians of preclinical and clinical groups can rely on MONOLIX for population analysis and to model PK/PD and other complex biochemical and physiological processes. MONOLIX is an easy, fast and powerful tool for parameter estimation in non-linear mixed effect models, model diagnosis and assessment, and advanced graphical representation.

MONOLIX is the result of a ten years research program in statistics and modeling, led by INRIA (Institut National de la Recherche en Informatique et Automatique) on non-linear mixed effect models for advanced population analysis, PK/PD, pre-clinical and clinical trial modeling & simulation.

MONOLIX is based on the MATLAB scientific environment published by Mathworks. MONOLIX is also available as a full-featured standalone software compiled with MATLAB libraries, and therefore does not require to purchase MATLAB licences.

Contents

1 Overview	4
2 Data	5
2.1 Declare data path and file - Examples	5
2.2 Declare headers - Examples	5
2.3 Examples and links with graphical interface	7
3 Variables declaration	8
3.1 Example 1 - basic use	9
3.2 Example 2 - covariate transformations	9
3.3 Example 3 - latent covariates	10
4 Individual parameters	12
4.1 Set up the distribution	12
5 Population parameters	16
6 Correlation structure	18
7 Structural model	20
8 Observations model	22
9 TASKS	24
9.1 estimatePopulationParameter	25
9.2 estimateFisherInformationMatrix	28
9.3 estimateIndividualParameter	28
9.4 estimateLogLikelihood	28
9.5 displayGraphics	28
9.6 Link with graphical interface	29
10 Comment your code	30

1 Overview

MLXTRAN is a declarative, human-readable language designed for complex models, including PKPD models.

At a high level, MLXTRAN is structured with sections:

- **DESCRIPTION**: a text describing the project
- **DATA**: all data-related informations
- **VARIABLES**: all informations related to the variables and their transformations
- **INDIVIDUAL**: statistical model of the individual parameters
- **POPULATION**: statistical model of the population parameters
- **CORRELATION**: correlation structure
- **STRUCTURAL_MODEL**: external file containing structural model
- **OBSERVATIONS**: statistical model of the observation model
- **TASKS**: workflow, algorithms settings, graphics settings

2 Data

The section ‘DATA’ enables to manage all the information provided by the data file and to link the data and the statistical model.

Below is the generic grammar of the section:

```
path=<data path>
file=<data file>,
headers={<data header>}
```

- path: directory path containing the data file. A macro %MLXPROJECT% is available and provides the path where the project MLXTRAN is saved.
- file: file containing the data
- headers: definition of data column functionalities

2.1 Declare data path and file - Examples

path=%MLXPROJECT%, file=Emax_errorband_data.txt,	The file ‘Emax_errorband_data.txt’ is in the directory of the current MONOLIX project
path=%MLXPROJECT%/data, file=Emax_errorband_data.txt,	The file ‘Emax_errorband_data.txt’ is in the sub-directory ‘data’ of the current MONOLIX project
path=/home/gandalf/mydata, file=Emax_errorband_data.txt,	The file ‘Emax_errorband_data.txt’ is in the directory ‘/home/gandalf/mydata’

2.2 Declare headers - Examples

Example 1 (phenobarbital_data.txt)

ID	TIME	DOSE	WEIGHT	APGAR	CONC
1	0	25	1.4	7	.
1	2	.	1.4	7	17.3
1	12	12.5	1.4	7	.
...
9	130.6	3.2	1.4	8	.
9	142.1	.	1.4	8	34.2
9	142.6	3.2	1.4	8	.
...

headers= { ID,TIME,Y,DOSE,IGNORE }

- ID, TIME and DOSE are keywords used by MONOLIX
- the column 'DV' is the observations 'Y'
- Finally the column 'GROUP' is ignored (IGNORE)

Example 2 (PKmixt_data.txt)

ID	TIME	DV	AMT	GROUP
1	0	.	10000	1
1	0.25	65.67216	.	1
1	0.5	91.84286	.	1
...
2	2	91.84286	.	0
2	2.5	91.842866	.	0
2	3	109.9155	.	0
...

headers= { ID,TIME,Y,DOSE,IGNORE }

- ID, TIME and DOSE are keywords used by MONOLIX
- the column 'DV' contains the observations 'Y'
- Finally the column 'GROUP' is ignored (IGNORE)

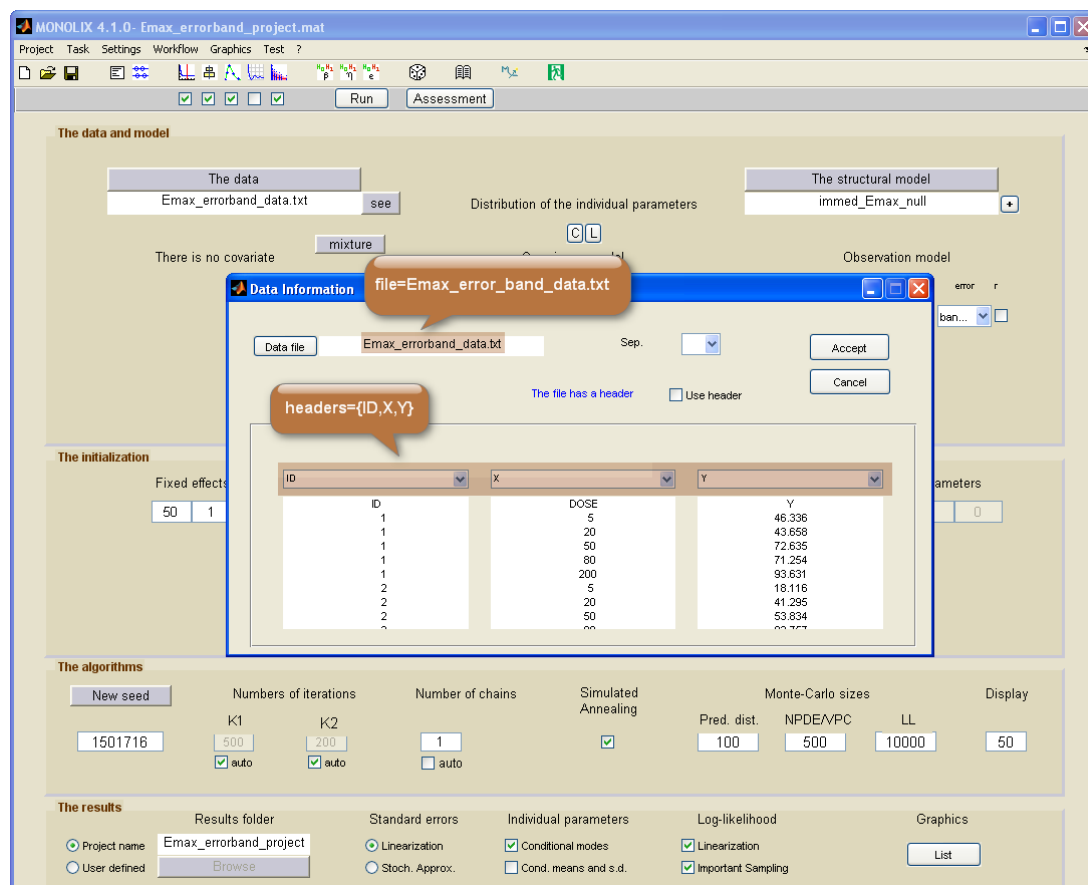
2.3 Examples and links with graphical interface

```
DATA:
path=%MLXPROJECT%,
file=Emax_errorband_data.txt,
headers={ID,X,Y},
```

The data file is 'Emax_errorband_data.txt'. It is stored in the directory of the MONOLIX project (%MLXPROJECT%). The header is:

- ID: subject id,
- X a regression variable
- Y the observation

In the MONOLIX Interface:



3 Variables declaration

The variable declaration links the data with the statistical model.

Below is the generic grammar of the section:

```

<column name> [ use=<cov>, type=<cat|latent>, (numberOfCategories=<value>) ],
<continuous transformation>=<formulae> [
    centeredBy=<median|mean|constant>,
    use=<cov>
],
<categorical transformation>= transform(
    <column name>,
    <group name> =<group value>,
    <group name> =<group value>,
    ...
) [use= <cov>, type= <cat>]

```

- **<column name>** is the column name as declared in the dataset (e.g. WEIGHT, SEX)
- **<continuous transformation>** means a continuous transformation of a column of the dataset. This column has to be filled with numbers (e.g. WEIGHT).
- **<categorical transformation>** means a transformation of a column of the dataset.
- **column** is the name of the dataset column (e.g. SEX)
- **name** is the new name of the group
- **value** is the value (set of values) associated with the group
- **use=cov** means that this column is used as a covariate. By default the covariate is assumed to be a continuous covariate
- **type=cat** means that the covariate is a categorical covariate
- **type=latent** means the covariate is not present in the data but is a latent covariate (assuming a mixture model)
- **numberOfCategories** is the number of categories of the latent covariate.

3.1 Example 1 - basic use

```
VARIABLES:
...
SEX [use=cov, type=cat], ;'SEX' is a column of the of dataset and
;is used as categorical covariate.

WEIGHT [use=cov], ;'WEIGHT' is a column of the of dataset
;and is used as continuous covariate.
```

3.2 Example 2 - covariate transformations

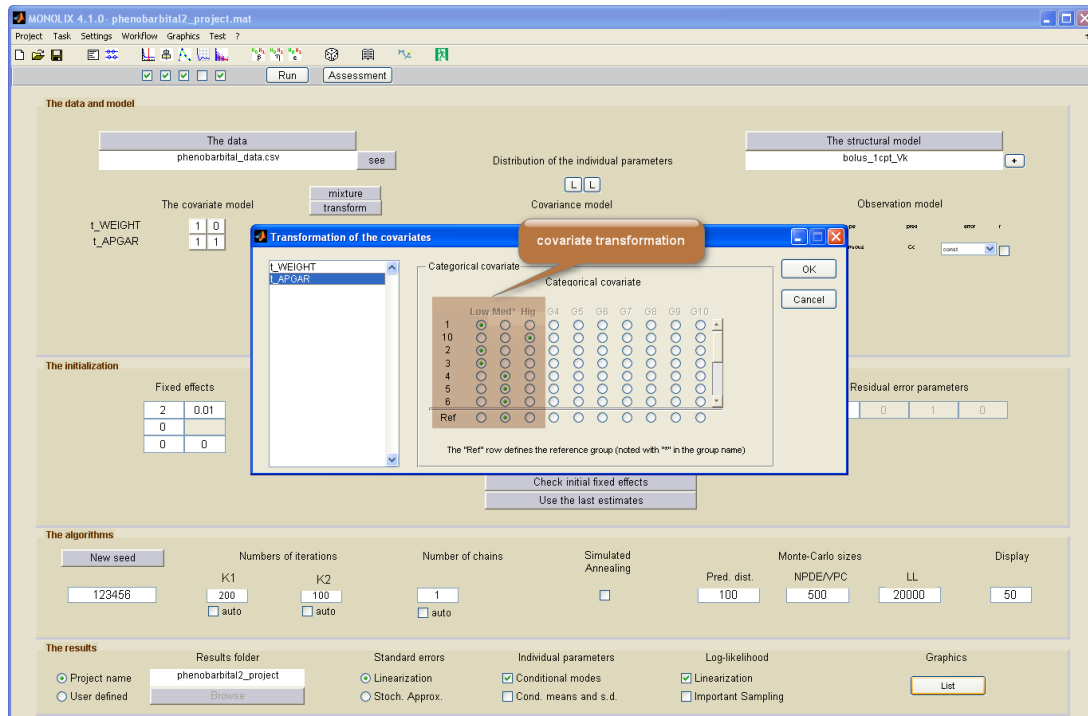
Data file extract:

ID	TIME	DOSE	WEIGHT	APGAR	CONC
1	0	25	1.4	7	.
1	2	.	1.4	7	17.3
1	12	12.5	1.4	7	.
...
9	130.6	3.2	1.4	8	.
9	142.1	.	1.4	8	34.2
9	142.6	3.2	1.4	8	.
...

```
...
VARIABLES:
...
tAPGAR=transform(
    {"APGAR",
    Low={1,2,3},
    Medium={4,5,6,7},
    High={10,9,8}
    }) [use=cov, type=cat],
WEIGHT,
t_WEIGHT=log(WEIGHT) [use=cov, centeredBy=median]
```

The categorical covariate '**t_APGAR**' is a transformation of the categorical covariate '**APGAR**'. The values taken by '**APGAR**' are grouped in 3 categories ('**Low**', '**Medium**' and '**High**').

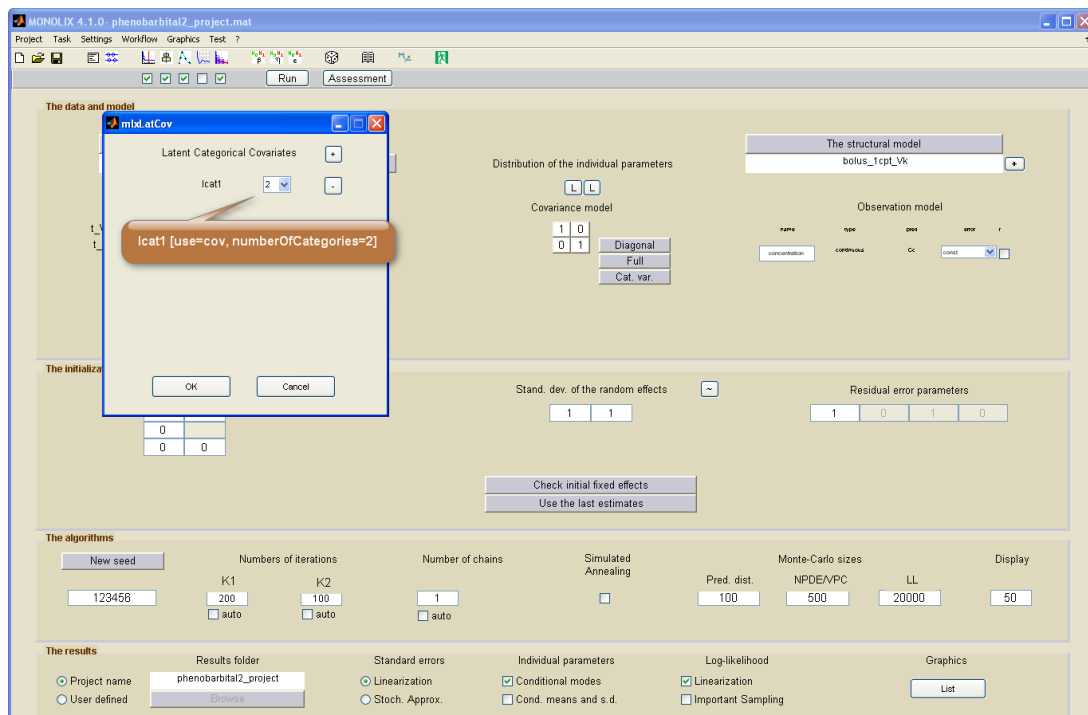
In the graphical interface



3.3 Example 3 - latent covariates

```
...
VARIABLES:
...
lcat1 [use=cov, type=latent, numberOfCategories=2],
...
```

In the graphical interface



4 Individual parameters

```

default      = { distribution = <lognormal|logitNormal|powerNormal|Normal|probitNormal>,
                  iiv = <yes|no> }
<parameter 1> = { iiv = <yes|no>,
                  iov = <yes|no>,
                  iov1 = <yes|no>,
                  ...
                  iov9 = <yes|no>,
                  covariate = { <cov1,cov2,...> },
                  cat_iiv = <cov>,
                  cat_iov1 = <cov>,
                  ...
                  cat_iov9 = <cov>,
                  distribution = <logNormal|logitNormal|powerNormal|Normal|probitNormal> },
...
<parameter_n>
...

```

- **default distribution** : default probability distribution of the individual parameters
- **iiv** : enable or disable inter-individual variability
- **iov / iov1 ... iov9**: enable or disable inter-occasion variability
- **covariate** : add covariate(s) in the statistical model
- **cat_iiv** : indicates that the variance of the random effect depends on a categorical covariate at IIV level
- **cat_iov/cat_iov1/...cat_iov9** : indicates that the variance of the random effect depends on a categorical covariate at IOV level
- **distribution** : sets the probability distribution of the parameter

4.1 Set up the distribution

It is possible to set up a default distribution (all parameters are assumed to have the same distribution) using the keyword `default` and/or set up a distribution for each individual parameter.

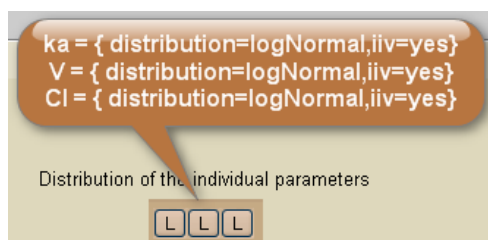
Example for a project with 3 individual parameters ka, V and Cl:
Equivalent notations

INDIVIDUAL:

```
default = {distribution=logNormal, iiv=yes},
ka,
V,
Cl
```

INDIVIDUAL:

```
ka = { distribution=logNormal, iiv=yes},
V = { distribution=logNormal, iiv=yes},
Cl = { distribution=logNormal, iiv=yes }
```



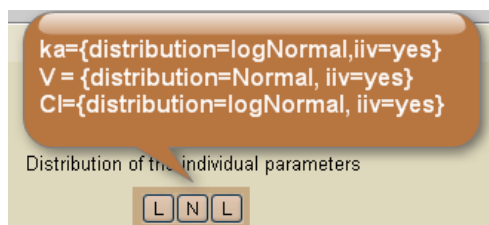
Equivalent notations

INDIVIDUAL:

```
default = {distribution=logNormal, iiv=yes},
ka,
V = { distribution=Normal, iiv=yes},
Cl
```

INDIVIDUAL:

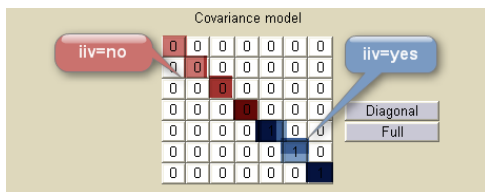
```
ka = { distribution=logNormal, iiv=yes},
V = { distribution=Normal, iiv=yes},
Cl = { distribution=logNormal, iiv=yes }
```



The available distributions are **Normal**, **logNormal**, **logitNormal**, **probitNormal** and **powerNormal**. The keywords **iiv**, **iov**, **iov2**, ..., **iov9**, allow to enable or disable variability on a specific level (**iiv** : inter-individual variability, **iov**: inter-occasion variability). The number after iov is the level of inter-occasion variability.

Example:

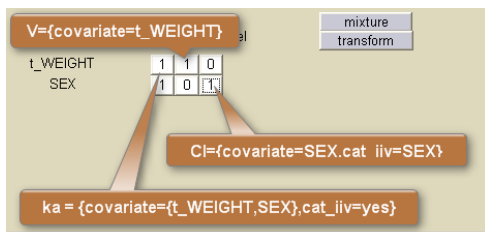
```
INDIVIDUAL:
Cl = { distribution=logNormal, iiv=no },
Tlag = { distribution=Normal, iiv=no },
V = { distribution=logNormal, iiv=no },
th1 = { distribution=logNormal, iiv=no },
th1 = { distribution=Normal, iiv=no },
th1 = { distribution=logNormal, iiv=no }
```



The keyword **covariate** is used to add a covariate to the statistical model.

Example (using the demos theophylline2)

```
INDIVIDUAL:
default = { distribution=logNormal, iiv=yes },
ka = { covariate=t_WEIGHT,SEX, cat_iiv=SEX },
V = { covariate=t_WEIGHT },
Cl = { covariate=SEX, cat_iiv=SEX }
```



To add dependencies on a categorical covariate, use the keyword **cat_iiv** at IIV level, **cat_iov** at IOV level and **cat_iov2**, **cat_iov3**, ... **cat_iov9** for the other IOV levels.

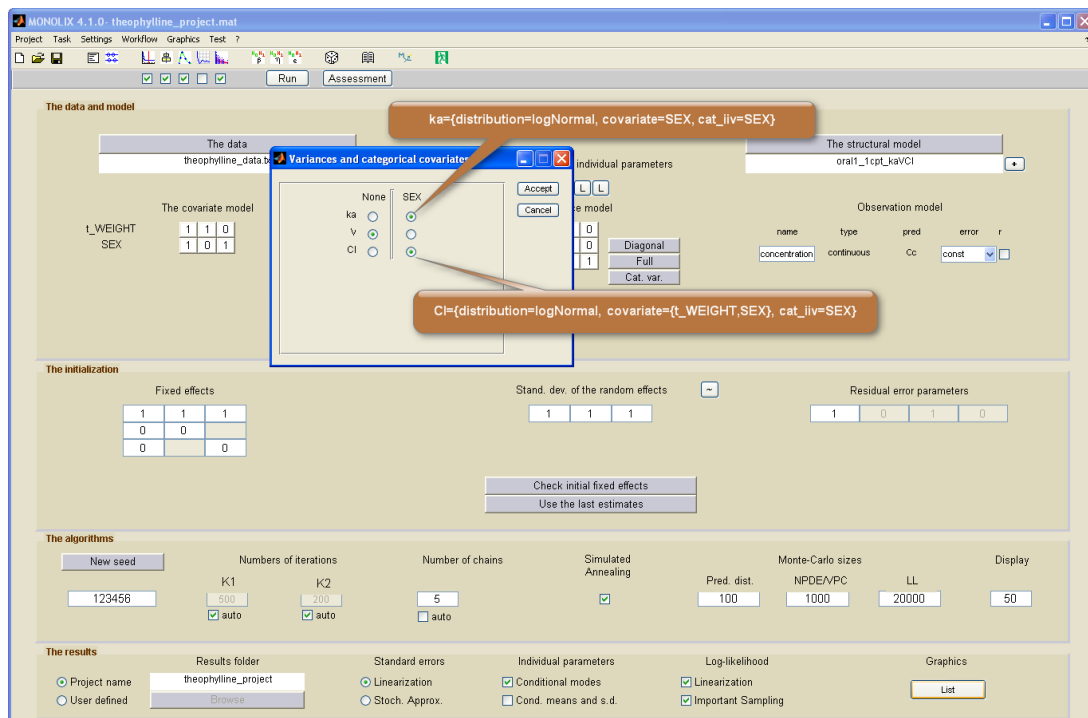
Example

INDIVIDUAL:

```

C1 = { distribution=logNormal,
       covariate={t_WEIGHT,SEX},
       iiv=yes,
       cat_iiv=SEX },
V = { distribution=logNormal, covariate={t_WEIGHT,SEX}, iiv=yes },
ka = { distribution=logNormal,
       covariate={t_WEIGHT,SEX},
       iiv=yes,
       cat_iiv=SEX }

```



5 Population parameters

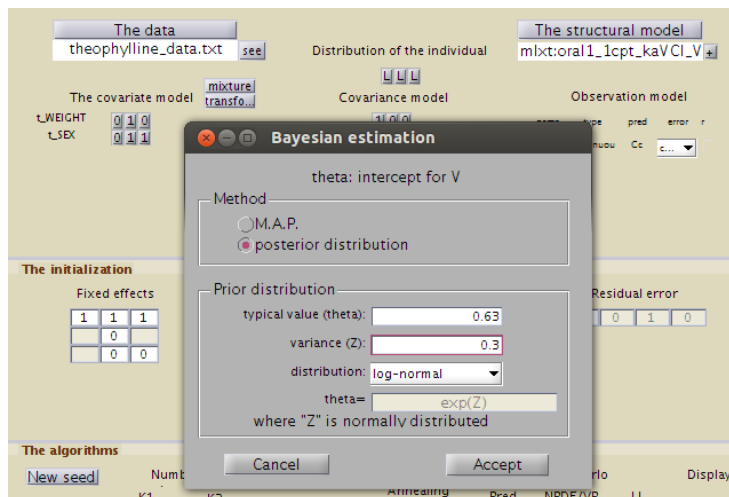
```

pop_{<parameter i>} = {
    distribution=<logNormal|logitNormal|Normal|probitNormal>,
    median=<real value>,
    variance|standardDeviation=<real value>
},
beta_{<parameter j,covariate k>} = {
    distribution=<logNormal|logitNormal|Normal|probitNormal>,
    median=<real value>,
    variance|standardDeviation=<real value>
},
...
pop_{<parameter_n>}
beta_{<parameter_p,covariate_m>}
...

```

- **distribution** : sets the probability distribution of the parameter
- **median** : typical value of parameter
- **variance|standardDeviation** : variance or standardDeviation of parameter

Example for a project with 3 individual parameters ka, V and Cl and a covariate W:



POPULATION:

```
pop_{ka} = { distribution=logNormal, median=0.98, variance=0.2 },  
pop_{V} = { distribution=logNormal, median=0.63, variance=0.3 } ,  
beta_{Cl,W} = { distribution=logNormal, median=0.3, standardDeviation=1.44 }
```

6 Correlation structure

Below is the generic grammar of the section:

```
correlationIIV = { {block1}, {block2}, ... {blockN}},  
correlationIOV = { {block1}, {block2}, ... {blockN}},  
correlationIOV2 = { {block1}, {block2}, ... {blockN}}},  
...  
correlationIOV9 = { {block1}, {block2}, ... {blockN}},
```

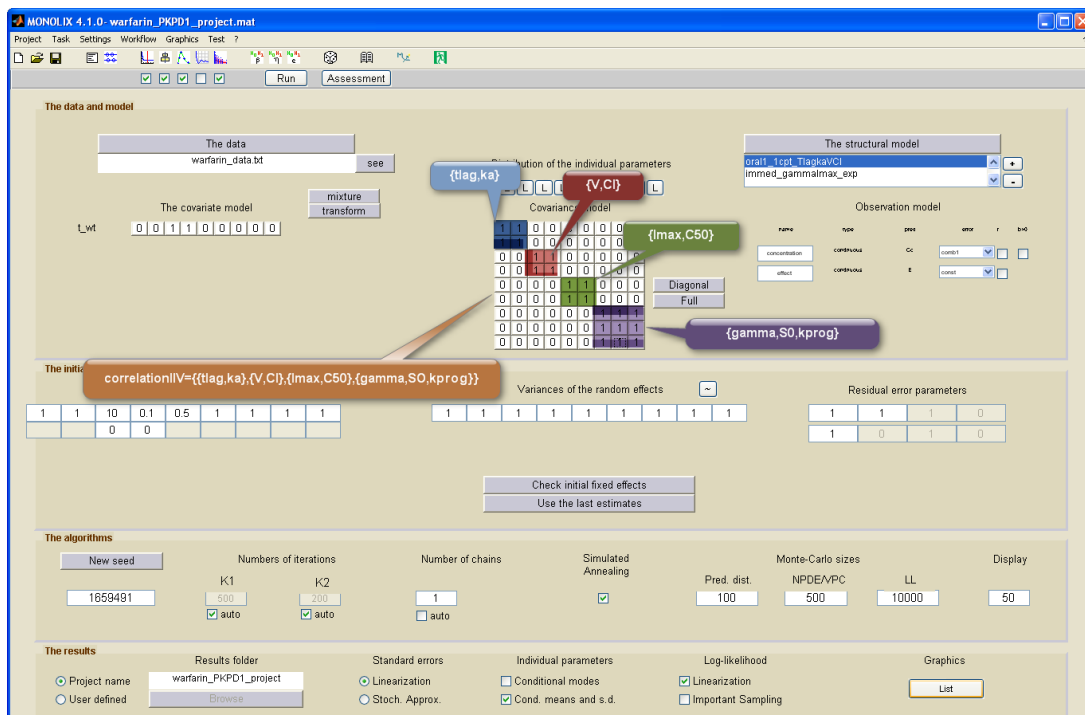
The covariance model of the random effects assumes a block matrix, where a block is a list of parameters.

Example (modified version of warfarin_PKPD1)

4 blocks are created:

- tlag,ka
- V,Cl
- Imax,C50
- gamma,S0,kprog

CORRELATION:
correlationIIV={{tlag,ka},
 {V,Cl},
 {Imax,C50},
 {gamma,S0,kprog}}



MONOLIX 4.1.0 - warfarin_PKPD1_project.mat

Project Task Settings Workflow Graphics Test ?

Run Assessment

The data and model

The data: warfarin_data.bt

The covariate model: L_{wt} 0 0 1 1 1 0 0 0 0 0

mixture transform

The structural model: oral1_1cpt_TlagkaVC1, immed_gammalmax_exp

Observation model: concentration, effect

Covariance model: Diagonal, Full

Variances of the random effects: 1 1 1 1 1 1 1 1 1 1

Residual error parameters: 1 1 1 0, 1 0 1 0

Check initial fixed effects, Use the last estimates

The algorithms

New seed: 1659491

Numbers of iterations: K1 500, K2 200

Number of chains: 1

Simulated Annealing: ☒

Pred. dist.: 100

Monte-Carlo sizes: NPDE/PC 500, LL 10000

Display: 50

The results

Results folder: warfarin_PKPD1_project

Standard errors: Linearization, Stoch. Approx.

Individual parameters: Conditional modes, Cond. means and s.d.

Log-likelihood: Linearization, Important Sampling

Graphics: List

correlationIV={{tlag,ka},{V,C1},{lmax,C50},{gamma,S0,kprog}}

7 Structural model

Below is the generic grammar of the section:

- Case of one structural model

```
STRUCTURAL_MODEL:
file=<structural model file without extension>,
path=<structural model path>,
output={<structural model output name>}
```

- Case of a serial structural model

```
STRUCTURAL_MODEL:
{
file=<structural model file without extension>,
path=<structural model path>,
output={<structural model output name>}
},
{
file=<structural model file without extension>,
path=<structural model path>,
output={<structural model output name>}
}
```

The structural model can be a MATLAB structural model or a MLXTRAN structural model script. To use libraries embedded in MONOLIX, it is recommended to use the %MLXPATH% macro that references the installation path of MONOLIX.

Example 1 (PK model)

STRUCTURAL_MODEL:
file=oral1_1cpt_kaVCl,
path=%MLXPATH%/PKLibrary,
output={f_1}

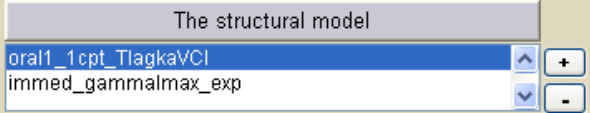
The structural model

oral0_1cpt_Tk0VCl

+

Example 2 (serial PK / PD models)

```
STRUCTURAL_MODEL:
{
    file=oral1_1cpt_TlagkaVCl,
    path=%MLXPATH% /libraries/PKLibrary,
    output={f_1}
},
{
    file=immed_gammaImax_exp,
    path=%MLXPATH%/libraries/PDLibrary,
    output={f_2}
}
```



8 Observations model

Below is the generic grammar of the section:

OBSERVATIONS:

```
<observation name> =  
{  
    type=<observation type>,  
    prediction=<prediction name>,  
    error=<error model>,  
    autocorrelation=<yes|no>  
}
```

- **<observation name>** : name of observation
- **type** : type of prediction (continuous, count, ...)
- **<prediction name>** : name of prediction (output of structural model)
- **<error model>** : error model applied on the output. The error model can be
 - constant,
 - proportional,
 - combined1,
 - combined2,
 - proportionalc,
 - exponential,
 - combined1c,
 - combined2c,
 - logit,
 - band(0,10),
 - band(0,100)

Example 1

STRUCTURAL_MODEL:

```
file=mlxt:turnover2_mlxt,  
path=%MLXPATH%/libraries/myLibrary/MLXTRAN,  
output={f_1,f_2}
```

OBSERVATIONS:

```
y_1 = { type=continuous, prediction=f_1, error=combined1 },  
y_2 = { type=continuous, prediction=f_2 error=constant }
```

Example 2

STRUCTURAL_MODEL:

```
file=mlxt:oral1_categorical_mlxt,  
path=%MLXPATH%/libraries/myLibrary/MLXTRAN,  
output={f_1,ll_2}
```

OBSERVATIONS:

```
y_1 = { type=continuous, prediction=f_1, error=combined1 },  
y_2 = { type=count, prediction=ll_2 }
```

9 TASKS

Below is the generic grammar of the section: [TASKS](#):

```
estimatePopulationParameters( initialValues={
    pop_<parameter> = <expression>,
    beta_<cov>_<parameter> =<expression>,
    a_<observation> =<expression>,
    b_<observation> = <expression>,
    c_<observation> = <expression>,
    r_<observation> = <expression>,
    omega_<parameter> = <expression>,
    (or omega2_<parameter> = <expression>,)
    gamma_<level>_<cov>_<parameter> = <expression>,
    (or gamma2_<level>_<cov>_<parameter> = <expression>,)
})
```

where

<expression> = <value> [method=MLE]

or <expression> = {**initialization**=<value>, **positive**=<yes|no>} [method=MLE] (use for ‘b’ error model parameter),

or <expression> = <value> [method=FIXED]

or <expression> = <value> [method=MAP]

or <expression> = <value> [method=BAYES]

for **MAP** or **BAYES** method it is always possible to define prior as follow (in this case the values set in the section **POPULATION _ PARAMETER** are ignored):

<expression> = { **prior**=<value>, **variance**=<value>} [method=MAP]

or <expression> = { **prior**=<value>, **standardDeviation**=<value>} [method=MAP]

or <expression> = { **prior**=<value>, **standardDeviation**=<value>} [method=BAYES]

or <expression> = { **prior**=<value>, **variance**=<value>} [method=BAYES]

estimateFisherInformationMatrix(**method**={**linearization**|**stochasticApproximation**}),

estimateIndividualParameters(**method**={**conditionalDistribution**,**conditionalMode**}),

estimateLogLikelihood(**method**={**importantSampling**, **linearization**}),

displayGraphics(**list**={**allg**, **projectSummary**, **spaghetti**, **individualFits**, **predVsObs**, **residuals**,
covariates, **vpc**, **npc**, **blq**, **vpcPredDist**, **categorizedData**, **distPsi**,
boxplot, **jointDist**, **cvSAEM**
}),


```
globalSettings={
    settingsGraphics=<path to graphics settings>,
    settingsAlgorithms=<path to algorithms settings>,
    resultFolder=<path to results>
    withVariance=<yes|no>
}
```

9.1 **estimatePopulationParameter** : run SAEM algorithm to estimate the population parameters

- Type of initialization:

- **pop_**{<parameter>}: initialization of population parameter <parameter>
- **beta_**{<cov>,<parameter>}: initialization of the effects of <cov> on <parameter>
- **a_**{<observation>}, **b_**{<observation>}, **c_**{<observation>} : initialization of error model parameters
- **r_**{<observation>} : autocorrelation parameter
- **omega_**{<parameter>}: standard deviation of random effect of parameter <parameter> on IIV level
- **omega2_**{<parameter>}: variance of random effect of parameter <parameter> on IIV level
- **gamma_**{<parameter>}: standard deviation of random effect of parameter <parameter> on IOV level
- **gamma2_**{<parameter>}: variance of random effect of parameter <parameter> on IOV level
- **gamma_**<level>_<parameter>}: standard deviation of random effect of parameter <parameter> on IOV level <level>
- **gamma2_**<level>_<parameter>}: variance of random effect of parameter <parameter> on IOV level <level>

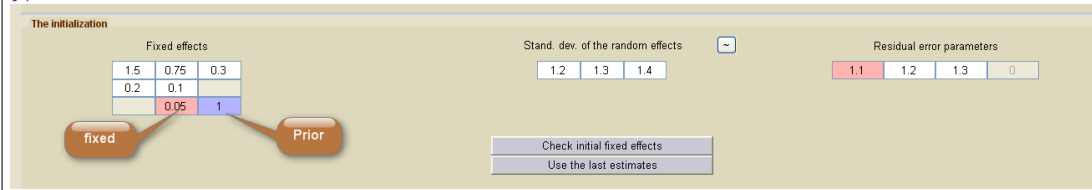
- Initialization value:

- **initialization**=<value> : value of initialization is <value>
- **fixed**: the value of parameter is fixed, otherwise the value will be estimated
- **standardDeviation**=<value> : standard deviation of initialization value used as prior
- **variance**=<value> : variance of initialization value used as prior
- **prior** : prior value

- Estimation method:
 - **[method=MLE]**: parameter is estimated with Maximum Likelihood Estimation
 - **[method=MAP]**: parameter is estimated with Maximum A Posteriori
 - **[method=BAYES]**: parameter is estimated with Bayesian Posterior
 - **[method=FIXED]**: parameter is fixed

Example 1

```
estimatePopulationParameters( initialValues={
  pop_{Cl} = 0.3,
  beta_{TREAT,Cl} = 0,
  beta_{SEX,Cl} = {prior=1, standardDeviation=0.25} [method=MAP],
  beta_OCC_Cl = 0,
  pop_{V} = 0.75,
  beta_{TREAT,V} = 0.1,
  beta_{S-TREAT,V} = 0.1,
  beta_{SEX,V} = 0.05 [method = FIXED],
  beta_{OCC,V} = 0,
  pop_{ka} = 1.5,
  beta_{TREAT,ka} = 0 [method = FIXED],
  beta_{S-TREAT,ka} = 0.2,
  beta_{OCC,ka} = 0,
  a_{concentration} = 1.1 [method = FIXED],
  b_{concentration} = 1.2,
  c_{concentration} = 1.3,
  r_{concentration} = 0,
  omega_{Cl} = 1.4 [method = FIXED],
  omega_{V} = 1.3,
  omega_{ka} = 1.2
})
```



The initialization window displays three main sections:

- Fixed effects:** A table with values for various parameters. The 'prior' value for $\beta_{SEX,Cl}$ is highlighted in blue, and the 'fixed' value for $\beta_{SEX,V}$ is highlighted in red.

1.5	0.75	0.3
0.2	0.1	
	0.05	1
- Stand. dev. of the random effects:** A row of input fields with values 1.2, 1.3, and 1.4.
- Residual error parameters:** A row of input fields with values 1.1, 1.2, 1.3, and 0.

Buttons at the bottom include 'fixed', 'prior', 'Check initial fixed effects', and 'Use the last estimates'.

Using the graphical interface (a modified IOV1_project from Demos directory)

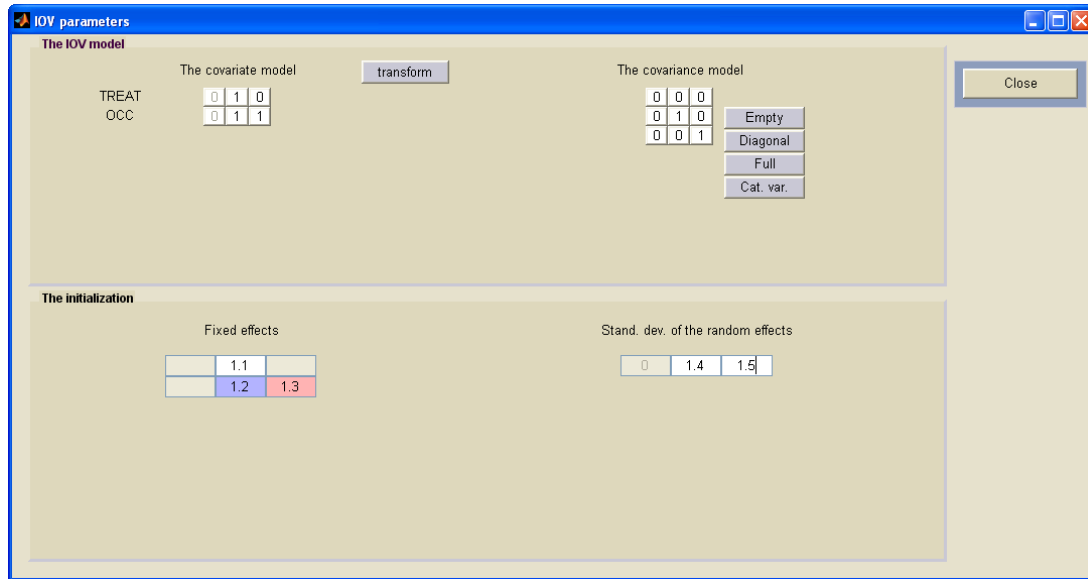
- 'Fixed effects' block

- first line is pop_ka, pop_V, pop_Cl initializations
- second line is beta_S-TREAT_ka, beta_S-TREAT_V initializations
- third line is beta_SEX_V, beta_SEX_Cl initializations
- ‘Stand. dev of the random’ are
 - omega_{ka},
 - omega_{V},
 - omega_{Cl}
- finally ‘Residual error parameters’ are
 - a_{concentration},
 - b_{concentration},
 - c_{concentration},
 - r_{concentration}

Example 2(show IOV initializations)

```
estimatePopulationParameters( initialValues={
    pop_{Cl} =0.05,
    beta_{SEX,Cl} =0,
    beta_{OCC,Cl} =1.3,
    pop_{V} =0.5,
    beta_{TREAT,V} =1.1 [method=FIXED],
    beta_{S-TREAT,V} =0,
    beta_{SEX,V} =0,
    beta_{OCC,V} = {prior=1.2, standardDeviation=0.25} [method=MAP],
    pop_{ka} =1.5,
    beta_{S-TREAT,ka} = 0,
    a_{concentration} = 1,
    b_{concentration} = { initialization = 0.3, positive = yes },
    omega_{Cl} =1,
    gamma_{Cl} =1.5,
    gamma_{V} =1.4,
    omega_{ka} =1
})
```

Using the graphical interface (using a modified IOV1_project from Demos directory)



IOV parameters

The IOV model

transform

TREAT
OCC

0	1	0
0	1	1

The covariance model

0	0	0
0	1	0
0	0	1

Empty
Diagonal
Full
Cat. var.

The initialization

Fixed effects

1.1	1.2	1.3
-----	-----	-----

Stand. dev. of the random effects

0	1.4	1.5
---	-----	-----

Close

9.2 `estimateFisherInformationMatrix`

Estimate the Fisher information matrix using linearization and/or stochastic approximation

9.3 `estimateIndividualParameter`

Estimate the individual parameters using the conditional mode and/or the conditional Mean (default: conditional mode)

9.4 `estimateLogLikelihood`

Estimate the log-likelihood by important sampling and/or by linearization (default: linearization)

9.5 `displayGraphics`

- `settingsGraphics`: path to the graphics settings XML file (not required, default settings can be used)
- `settingsAlgorithms`: path to the algorithms settings XML file (not required, default settings can be used)

- withVariance: the initialization values are expressed with variance of parameters

9.6 Link with graphical interface

Example 1

TASKS:

```
estimatePopulationParameters(),
estimateFisherInformationMatrix( method={ linearization }),
estimateIndividualParameters( method={conditionalDistribution}),
estimateLogLikelihood( method={importantSampling})
```

The results		Results folder	Standard errors	Individual parameters	Log-likelihood	Graphics
<input checked="" type="radio"/> Project name	theophylline2_project	<input checked="" type="radio"/> Linearization	<input type="checkbox"/> Conditional modes	<input type="checkbox"/> Linearization	<input type="checkbox"/> Important Sampling	<input type="button" value="List"/>
<input type="radio"/> User defined	<input type="button" value="Browse"/>	<input type="radio"/> Stoch. Approx.	<input checked="" type="checkbox"/> Cond. means and s.d.	<input checked="" type="checkbox"/> Important Sampling		

Example 2

TASKS:

```
estimatePopulationParameters(),
estimateFisherInformationMatrix( method={ stochasticApproximation }),
estimateIndividualParameters( method={conditionalMode}),
estimateLogLikelihood( method={linearization})
```

The results		Results folder	Standard errors	Individual parameters	Log-likelihood	Graphics
<input checked="" type="radio"/> Project name	theophylline2_project	<input type="radio"/> Linearization	<input checked="" type="checkbox"/> Conditional modes	<input checked="" type="checkbox"/> Linearization	<input type="checkbox"/> Important Sampling	<input type="button" value="List"/>
<input type="radio"/> User defined	<input type="button" value="Browse"/>	<input checked="" type="radio"/> Stoch. Approx.	<input type="checkbox"/> Cond. means and s.d.	<input type="checkbox"/> Important Sampling		

10 Comment your code

MLXTRAN allows to comment the code with the ';' character. The text after ';' is ignored until the end of the line.

Example

DESCRIPTION:Project description

DATA:

path=%MLXPROJECT%, ;put comment here

file=theophylline_data.txt,

; WARNING the following code is ignored:headers={ID,DOSE,TIME,Y,COV,CAT},