LIXOFT

MONOLIX

Version 4.2.2

MARCH 2013

A software for the analysis of nonlinear mixed effects models

Maximum likelihood estimation

Model selection

Hypothesis testing

Graphical analysis

Data simulation

...

IMPORTANCE SAMPLING
STOCHASTIC
MCMC
SIMULATED ANNEALING
SAEM
HMM
METROPOLIS

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

Chapter 1

Introduction

1.1 The objectives

The objectives of Monolix are to perform:

1. Parameter estimation for nonlinear mixed effects models

    - computing the maximum likelihood estimator of the population parameters, without any approximation of the model (linearization, quadrature approximation, ...), using the Stochastic Approximation Expectation Maximization (SAEM) algorithm,
    - computing standard errors for the maximum likelihood estimator
    - computing the conditional modes, the conditional means and the conditional standard deviations of the individual parameters, using the Hastings-Metropolis algorithm

2. Model selection

    - comparing several models using some information criteria (AIC, BIC)
    - testing hypotheses using the Likelihood Ratio Test
    - testing parameters using the Wald Test

3. Easy description of pharmacometric models (PK, PK-PD, discrete data) with the Mlxtran language

4. Goodness of fit plots

5. Data simulation.

Monolix handles a broad spectrum of models including models defined with differential equations, left censored data, discrete data models, repeated time to events, hidden Markov models, mixture models,. . .

Theoretical analysis of the algorithms used in this software can be found in [9, 10, 13, 14, 1]. Several application of SAEM in agronomy [19], animal breeding [12] and PKPD analysis [4, 17, 23, 25, 27, 28, 3, 29] have been published by several members of the Monolix group and other authors. Several applications to PKPD analysis and several extensions to complex models were also proposed during the last PAGE (Population Approach Group in Europe) meetings ([2, 16, 15, 20, 22, 24, 26, 8, 18, 7] as well as a comparison of estimation algorithms [11], (http://www.page-meeting.org).

The aim of the present User Guide is to help a Monolix beginner to discover the software abilities. Chapter 2 contains the installation guide, Chapter 3 explains how to use Monolix and its graphical interface. Advanced Features and examples are detailed in Chapter 4. And the use of the software without the graphical interface and batch mode is described in Chapter 5.

The statistical models handled by Monolix are given in Appendix A and a list of visual preferences for result graphics is given in Appendix B.

The user guide does not cover Mlxtran programming in details, and is therefore completed by two additional documents:

- `modelMLXTRANtutorial.pdf`   is a slidedeck tutorial on how to describe pharmacometric models with Mlxtran .

- `ProjectMLXTRAN.pdf` presents how to easily program and customize complete Monolix projets with Mlxtran .

Chapter 2

Installing and running Monolix®

2.1 Downloading packages

The Monolix packages can be downloaded through the download manager hosted at `http://download.lixoft.com`. The download manager is available for users provided with an access key. Different Monolix packages are available, depending on the Matlab version and of the operating system. Monolix currently supports Windows XP/Vista/Seven 32bits, Linux (all common distributions) 32/64 bits. On Windows XP/Vista/Seven 64 bits, Monolix standalone version can run in 32 bits mode.

Choice of Monolix versions

- Matlab versions:
  - Linux matlab-r2010b-r2011a-r2011b-r2012a (64 bits)
  - Linux matlab-r2009a-r2010a (32 bits)
  - Linux matlab-r2010b-r2011a-r2011b-r2012a (32 bits)
  - Linux matlab-r2009a-r2010a (64 bits)
  - Windows (seven and vista) matlab-r2009a-r2010a (32 bits)
  - Windows (seven and vista) matlab-r2010b (32 bits). Due to bugs in Matlab 2010b, it is strongly recommanded to use Matlab 2010b-SP1.

- Standalone versions:
  - Linux (32 bits)
  - Linux (64 bits)

    – Windows (32 bits)

    – Windows (64 bits)[1]

2.2 Installation

2.2.1 Prerequisites

perl is required to run perlScripts and the validation suite; it is not required otherwise.

Linux specifics

- install sharutils : uudecode is required to uncompress the MONOLIX package;

- make sure you have gcc/g++/make installed or install them.

Windows 64bits specifics

The 32 bits *standalone* version of MONOLIX runs fine on Windows 7 64bits. You will need to install the 64 bits Windows version of MONOLIX in any of these situations:

- On other 64 bits versions of Windows (non Windows 7);

- If you wish to use a MATLAB version of MONOLIX .

- If you simply prefer to use a 64bits version of standalone MONOLIX , although in practice this should not have an impact on the performance.

The installer of the 64 bits Windows version of MONOLIX executes the Windows SDK installer. This SDK embeds the C++ compiler required to generate MLXTRAN modules.The installation process is as follows:

---

[1]MONOLIX will run in 32 bits mode
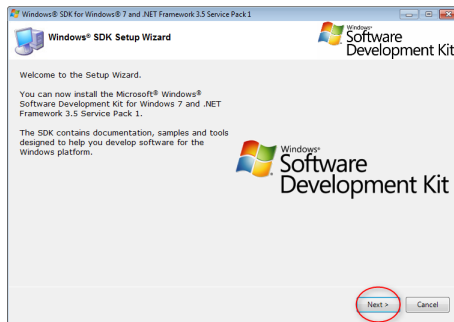
Welcome page of the SDK installer: click on 'next' button to continue;
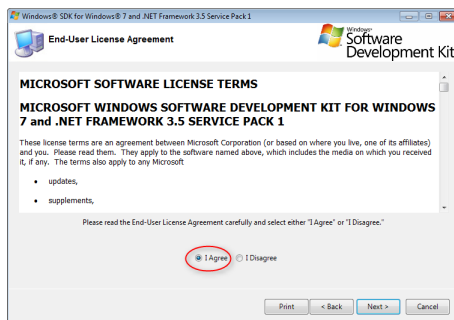
License agreement page: accept the agreement by selecting 'I agree', then click on 'next' button to continue;
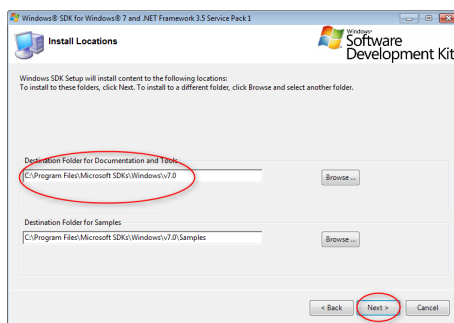
Choose the installation path. The proposed directories are required by MONOLIX ;
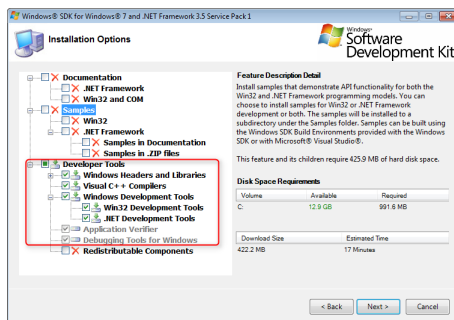
The component used by MONOLIX are the compiler only, therefore it is not necessary to install documentation and samples.
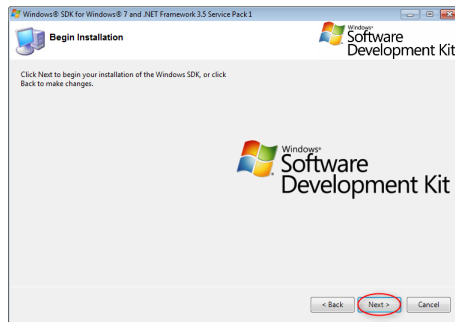
On 'Begin installation' page, click on 'next' button to continue;
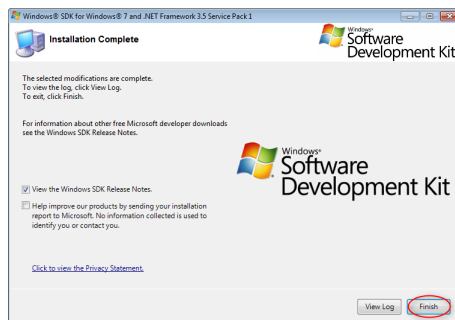
On 'Installation complete' page click on 'finish' button to continue;

After completion of the SDK installation procedure, the Monolix installer resumes the Monolix installation procedure.

If the SDK was already installed on the computer, the SDK installer will propose a list of actions:
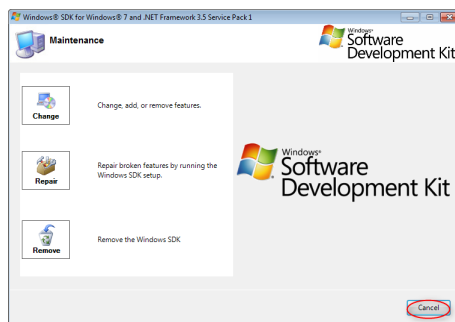
No action is necessary, click on 'Cancel';
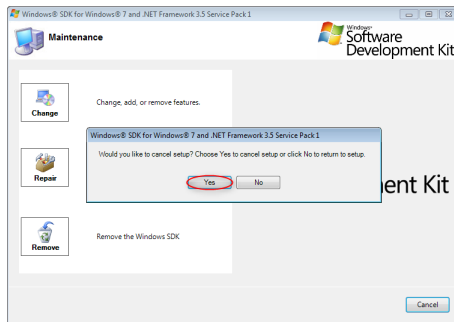
Confirm the 'Cancel' choice by clicking on 'Yes' button.

2.2.2   About Installer

- Linux : the installer is a self-extractable archive.

  - run the following command (depending on your os version):
    ```
    #> sh Monolix-4.2.2-matlab2010a-linux32.bin
    or
    #> sh Monolix-4.2.2-matlab2010bSP1-linux32.bin
    or
    #> sh Monolix-4.2.2-standalone2008b-linux32.bin
    or
    #> sh Monolix-4.2.2-matlab2010a-linux64.bin
    or
    #> sh Monolix-4.2.2-matlab2010bSP1-linux64.bin
    or
    #> sh Monolix-4.2.2-standalone2008b-linux64.bin
    ```

  - you can specify the target installation directory by giving the path as argument
  - a directory containing MONOLIX will be created in the directory installation path

- Windows

  - copy the installer on your Desktop or in your windows temporary directory
  - Double click on the executable and follow the instructions.

2.2.3   Directory structure

The MONOLIX directory structure is divided in two parts:

- the software directory containing the MONOLIX program,

- the personal user directory containing the MONOLIX workspace and documentation

LIXOFT

Installation directory

```
Monolix..................................................Monolix root directory
    └─ monolix413......................................Monolix version directory
         ├─ bin.......................................................tools directory
         ├─ config...............................................configuration files
         │    ├─ Demos................set of demos (copied in Monolix user directory)
         │    ├─ graphics........................................ graphics configurations
         │    │    ├─ listOfGraphics................. graphics predefined configurations
         │    │    ├─ project.............. graphics default configurations for Mlxtran
         │    │    └─ settings........................... graphics default configurations
         │    ├─ scenario.............................................. predefined scenarii
         │    └─ session................................. monolix session configuration
         ├─ factory ....................................................Mlxtran c++ API
         ├─ jar ........................................................Java library
         ├─ lib .............................................................C++ library
         ├─ matlab ...............................................Monolix main program
         │    ├─ libraires ..............................................models libraires
         │    ├─ mlxCore ...............Monolix core : all algorithms (SAEM, FIM, ...)
         │    ├─ mlxDelegate ........glue to present Monolix project (HMI, batch, ...)
         │    ├─ mlxIO ...............input / output components (read .mat, .xmlx, ...)
         │    ├─ mlxMath ...................................misc mathematical functions
         │    ├─ mlxTools .....................................some tools (mat to xmlx)
         │    └─ mlxUseful ...........................................generic components
         ├─ perlScripts ..................................................Perl scripts
         ├─ reference ....................reference project for the validation suite
         └─ tools ...........................external tool used by Monolix (cmake)
```

User directory

The user directory is created after the first launch of Monolix. This directory contains the basic configuration of Monolix, documentation, demos, log files, license file, ...

```
monolixData...............................................Monolix root directory
    ├─ monolix413......................................Monolix version directory
    │    ├─ doc ..................................................Monolix documentation
    │    ├─ log...........................................................log files
    │    ├─ script_modules.........................compiled Mlxtran modules
    │    ├─ perlScripts ..............................................Perl scripts
    │    └─ work...............................................user working directory
    │         └─ Demos.................................................Modifiable demos
    ├─ license ......................................................tools directory
    └─ config.....................................................configuration files
```

LIXOFT

2.2.4   About Plugins

Monolix can embed the BiM plugin, a faster and more accurate ODE solver, not included by default due to its license restrictions. The plugin must therefore be downloaded and installed separately:

- Linux

    - libBim.tgz (using the command: `tar xfz libBim.tgz` or your graphical archiver)
    - copy the files stored in the directory libBim into the library directory of Monolix :
        * for the Matlab version : `<install path>/lib`
        * for the standalone version : `<install path>/bin/Monolix_mcr/runtime/lib`

- Windows : copy libBim.dll into the library directory of Monolix :

    - for the matlab version : `<install path>\lib`

    - for the standalone version : `<install path>\bin\Monolix_mcr\runtime\lib`
      With the standalone version of Monolix the directory `<install path>` is located at:
        * under Window XP or Windows Server 2003:
          `c:\Documents And Settings\All Users\Application Data`
        * under Window Vista, Windows 7 or Windows Server 2008:
          `c:\ProgramData`

      Important notice: these directories may be hidden by the operating system, thus you have to configure your file browser for access.

2.2.5   Running Monolix

- Linux

    - Matlab version
        * start Matlab
        * go to directory '`<install path>/matlab`' and type `monolix`.
    - Standalone version: go to '`<install path>/bin`' and type `./Monolix.sh`.

- Windows

– MATLAB version
   * start MATLAB
   * go to directory `<install path>\matlab`' and type `monolix`.
– Standalone version: go to '`<install path>\bin`' and type Monolix.bat.

2.2.6 Installation use cases

Desktop

MONOLIX is installed on the computer of the user and the user has a personal activation key (see Section 2.2.7 *Desktop license*). After the installation or during the first startup of MONOLIX a popup titled 'Lixoft Activate' appears and asks the activation key. When the activation procedure is finished, MONOLIX will be configured (typically a directory `monolixData` is created in the user home directory) and launched.

Desktop with a shared MONOLIX installation

MONOLIX is installed on a remote server and the user accesses to MONOLIX through a shared directory (via CIFS, Network drive, NFS, ...) and the user has a personal activation key (see Section 2.2.7 *Desktop license*).
During the first startup of MONOLIX a popup title 'Lixoft Activate' appears and asks the activation key. When the activation procedure is finished, MONOLIX will be configured (typically a directory `monolixData` is created in the user home directory) and launched.

Application server with a shared MONOLIX installation

MONOLIX is installed on a remote server using the procedure described in Section 2.2.7 'Floating license'. The license file (obtained during activation procedure) is copied in the directory

- `<monolix user install path>/config/system/access` for the MATLAB version of MONOLIX

- or `<monolix install path>/bin/Monolix_mcr/runtime/config/system/access` for the standalone version of MONOLIX .

The user accesses to MONOLIX through a shared directory (via CIFS, Network drive, NFS, ...). The user runs MONOLIX directly, no activation is required. Nevertheless, when a user runs

Monolix a license token is taken.

If all license tokens are used (too many users run Monolix in the same time), a popup titled 'Lixoft activate' appears and the user is supposed to wait until at least one token is released.

Application server with a remote connection

With a floating license   Monolix is installed on a remote server using the procedure described in Section 2.2.7 'Floating license'. The license file (obtained during activation procedure) is copied in the directory

- `<monolix user install path>/config/system/access` for the Matlab version of Monolix

- or `<monolix install path>/bin/Monolix_mcr/runtime/config/system/access` for the standalone version of Monolix .

The user accesses to Monolix using a remote desktop application.

The user runs Monolix directly, no activation is required. Nevertheless, when a user runs Monolix a license token is taken.

If all license tokens are used (too many users run Monolix in the same time), a popup titled 'Lixoft activate' appears and the user is supposed to wait until at least one token is released.

With desktop licenses   Monolix is installed on a remote server, the user accesses to Monolix using a remote desktop application and has a personal activation key (see Section 2.2.7 *Desktop license*).

During the first startup of Monolix a popup title 'Lixoft Activate' appears and asks the activation key. When the activation procedure is finished, Monolix will be configured (typically a directory `monolixData` is created in the user home directory) and launched.

Application server with a desktop installation

Monolix is installed on a remote server using the procedure described in Section 2.2.7 'Floating license'. Each Monolix user is supposed to have a copy of the license file obtained during the activation procedure. After the installation or during the first startup of Monolix , a popup titled 'Lixoft Activate' appears. The tab 'With License file' has to be selected. The user is supposed to browse to the copy of the license file to activate Monolix . When a user runs Monolix a license token is taken.

If all license tokens are used (too many users run Monolix in the same time), a popup titled 'Lixoft activate' appears and the user is supposed to wait until at least one token is released.

Cluster installation with a shared Monolix **installation**

Monolix is installed on a master server using the procedure described in Section 2.2.7 'Floating license'. The license file (obtained during activation procedure) is copied in the directory

- `<monolix user install path>/config/system/access` for the Matlab version of Monolix

- or `<monolix install path>/bin/Monolix_mcr/runtime/config/system/access` for the standalone version of Monolix .

Each cluster node accesses to Monolix through a shared directory (via CIFS, Network drive, NFS, ...).
The user runs Monolix directly, no activation is required. Nevertheless, when a user runs Monolix a license token is taken (there is no limit of runs on cluster nodes).
If all license tokens are used (too many users run Monolix in the same time), a popup titled 'Lixoft activate' appears and the user is supposed to wait until at least one token is released.

Cluster installation with Monolix **installed on each node**

License server (RLM) has to be installed on a master server and the license file is download using the procedure described in Section 2.2.7 'Floating license'. Monolix is installed on each cluster. During this installation it is not necessary to activate Monolix when the popup titled 'Lixoft activate' appears (just close the popup). The license file (obtained previously) is supposed copied in the directory

- `<monolix user install path>/config/system/access` for the Matlab version of Monolix

- or `<monolix install path>/bin/Monolix_mcr/runtime/config/system/access` for the standalone version of Monolix

of each node.

2.2.7   License

Monolix licenses can be of the following types:

- Individual license - named user. The named user can install and run Monolix on a predetermined number of different computers.

- Floating license - concurrent access. The license is hosted by a license server, and MONOLIX can either run on a server or individual workstations.

Remark: the former license management tool uses a license file (`license.ini`); this type of license is deprecated since MONOLIX *version 4.1.3.*

Desktop license

The activation key (provided by LIXOFT) must be entered in the dialog box titled 'LIXOFT license activation' ('With activation key' tab). This dialog box only appears when no license is available on the user's computer or when the license expires.
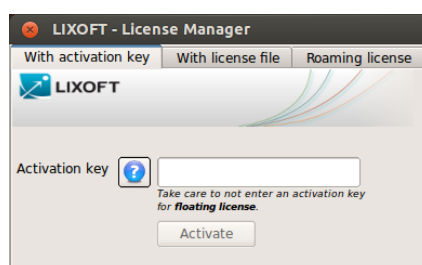
Floating license

The use of a floating license requires to set up a license server. In this case there are two installation strategies for MONOLIX users:

- install MONOLIX on a directory shared by all MONOLIX users,

- install MONOLIX on each user's computer and copy the license file obtained as described below into the directory:

  - `<monolix user install path>/config/system/access` for the MATLAB version of MONOLIX ,
  - or `<monolix install path>/bin/Monolix_mcr/runtime/config/system/access` for the standalone version of MONOLIX .

After the installation process, when the 'Lixoft activate window' appears just close the window (do not enter the activation key of the floating license). Then, start the RLM server, located at:

- `<monolix install path>/tools/rlm/rlm{.exe}` for the MATLAB version of MONOLIX ,

- or `<monolix install path>/bin/Monolix_mcr/runtime/tools/rlm/rlm{.exe}` for the standalone version of MONOLIX .

At this step there is no license available yet; the IT manager should use the RLM web server to download the license by following the procedure below:

1. In the web browser, type `<IP>:5054`, where <IP> is the IP address of the computer hosting the RLM server (e.g. `192.168.46.248:5054`).
   Notice that the RLM server opens two ports : 5053 and 5054. The first port (5053) is a service port used for the transactions of licenses. The second port (5054) is the RLM web server port used to access to the RLM configuration through a web browser.
   It is possible that one or both ports may have been used by another application.

   - If the web server port (5054) is not available you can launch RLM server with a new port by using the program option `-ws` (e.g: `rlm -ws 5055`) in this case, the access to RLM configuration through a web browser is done using the address `<IP>:<NEW PORT>` (e.g. 192.168.46.248:5055).

   - If the server port (5053) is not available, a file `config.conf` has to be created in the rlm directory and has to contain the following information:
     `HOST <IP> <MAC ADDRESS> <NEW PORT>`
     e.g.
     `HOST 192.168.46.245 a8c0f82e 5060`
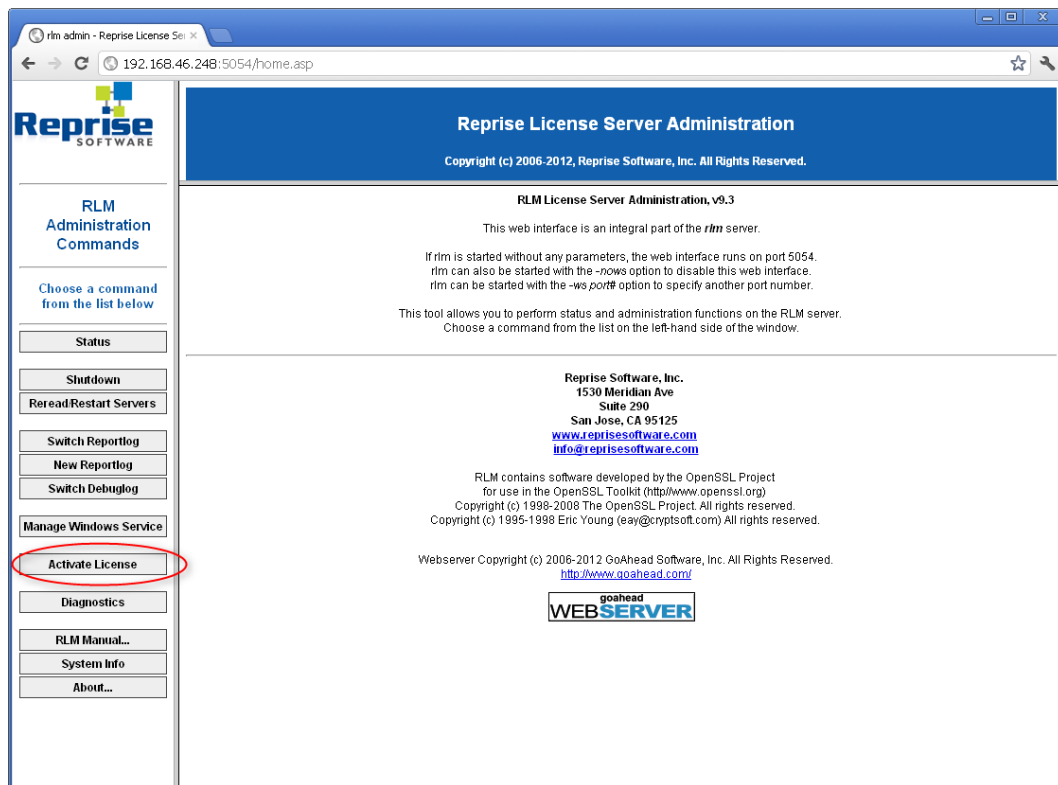
2. Begin license activation:
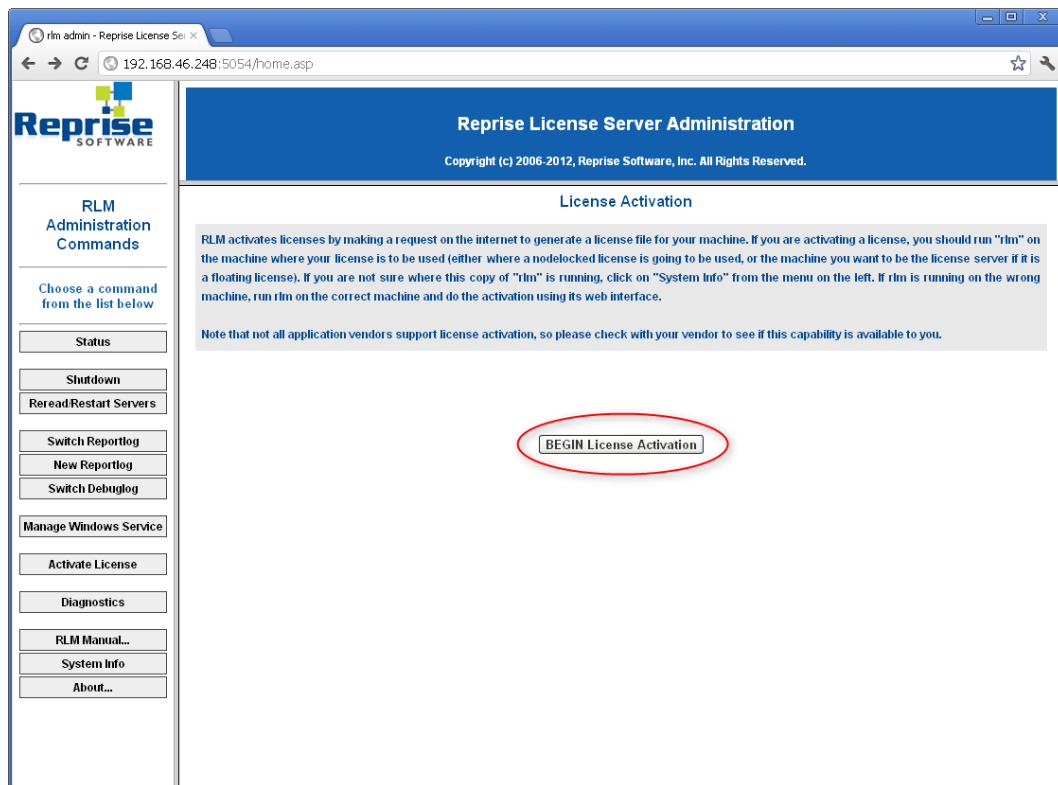
3. Enter the RLM activation url : `activate.lixoft.net`. And click on `Next` button.

   If the rlm server does not have Internet access, the license has to be created by Lixoft . Send a mail to `support@lixoft.com` with the following informations:

   - Mac address of the computer hosting the RLM server
   - IP address of the computer hosting the RLM server
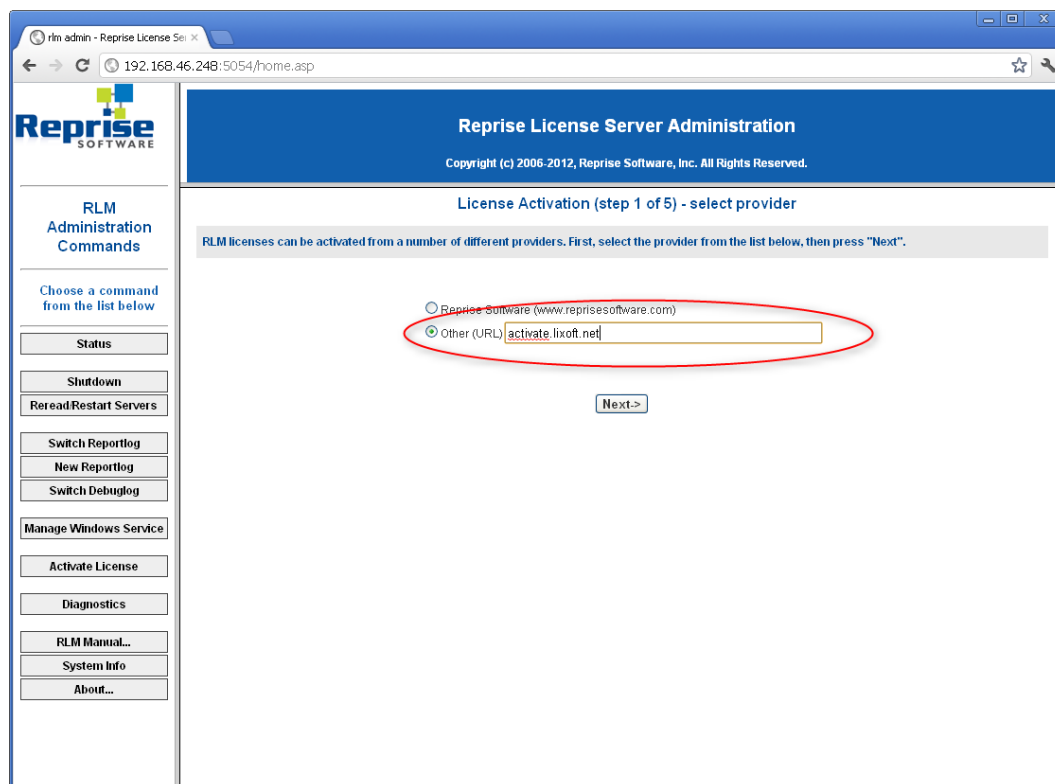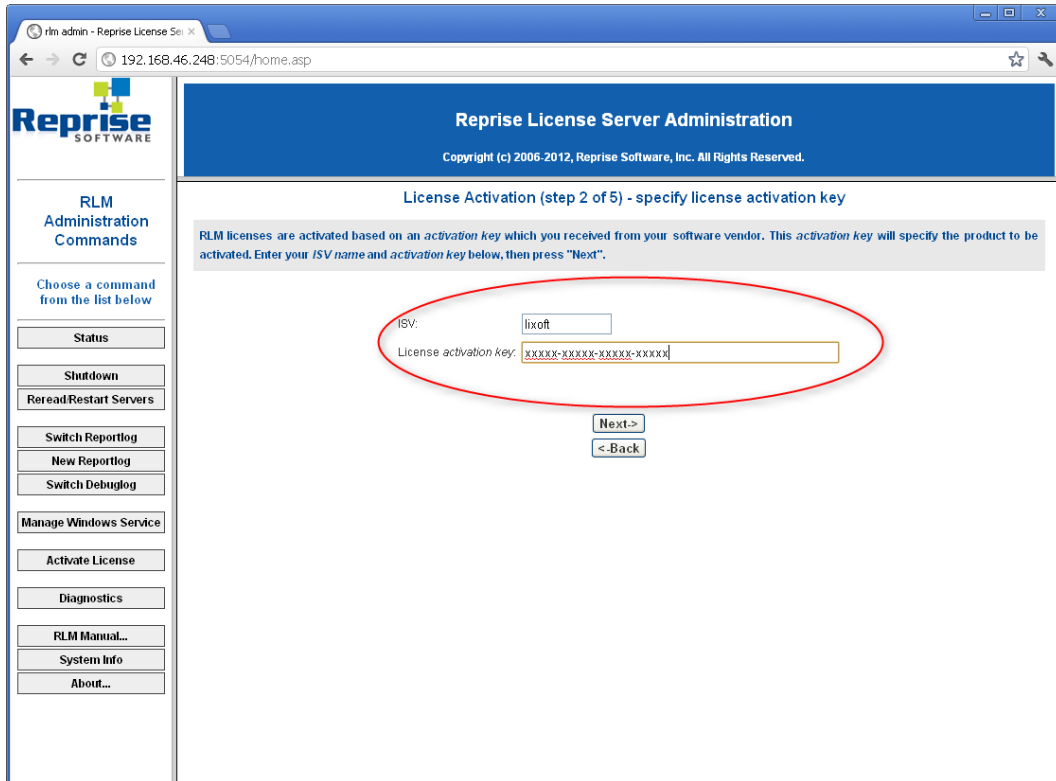
LIXOFT will send in return a '.lic' file which has to be copied in the directory

- <monolix install path>/config/system/access (MATLAB version of MONOLIX)
- <monolix install path>/bin/Monolix_mcr/runtime/config/system/access (standalone version of MONOLIX).

At this step, the installation of MONOLIX is complete.
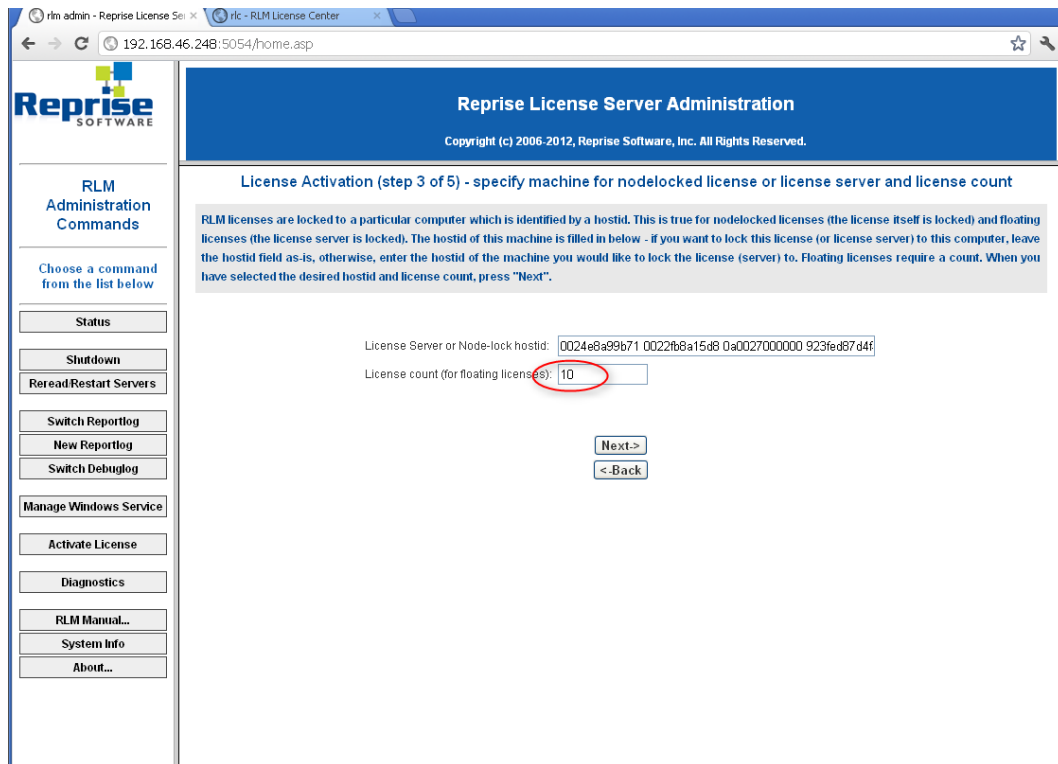
4. Activate the license.

   Fill the ISV input with the string 'lixoft' (without the quotes) and the License activation key with the activation key provided by LIXOFT (key format is xxxx-xxxx-xxxx-xxxx)

5. Enter (at maximum) the number of bought licenses, then click on `Next` button

   Notice, the number of licenses cannot exceed the number of bought licenses.

6. Select the license directory and file.

   In the field named `License file to create` write the full path to license file
   `<monolix install path>/config/system/access/myfloat.lic` for the MATLAB version
   of MONOLIX
   or `<monolix install path>/bin/Monolix_mcr/runtime/config/system/access` for the
   standalone version.
   e.g: if the MONOLIX (matlab version) installation directory is `/media/share/monolix` the
   input field name `License file to create` should contain
   `/media/share/monolix/config/access/myfloat.lic`

   This license file has to be copied on each installation of MONOLIX :

   - If Monolix is installed on a shared space (i.e. each node of the cluster has an access
     to this directory), copy the license file into the directory
     `<monolix install path>/config/system/access/` for the MATLAB version of MONOLIX
     or `<monolix install path>/bin/Monolix_mcr/runtime/config/system/access` for
     the standalone version.
     Make sure that the MONOLIX directory is accessible from each cluster node.

Example (with a Matlab **version of** Monolix **)**

- Monolix is installed on the computer `master-computer` in the directory:
  `/usr/local/monolix/`.
  The license is in the directory :
  `/usr/local/monolix/config/access/`
- The RLM server is run on the computer `master-computer`.
- Cluster computers mount the directory `/usr/local/monolix/`.
- Each monolix user runs Monolix from the previously mounted directory.

- If Monolix is installed on each node of the cluster, copy the license file on each computer in the directory `<monolix install path>/config/system/access` for the Matlab version of Monolix or
  `<monolix install path>/bin/Monolix_mcr/runtime/config/system/access` for the standalone version.

Example
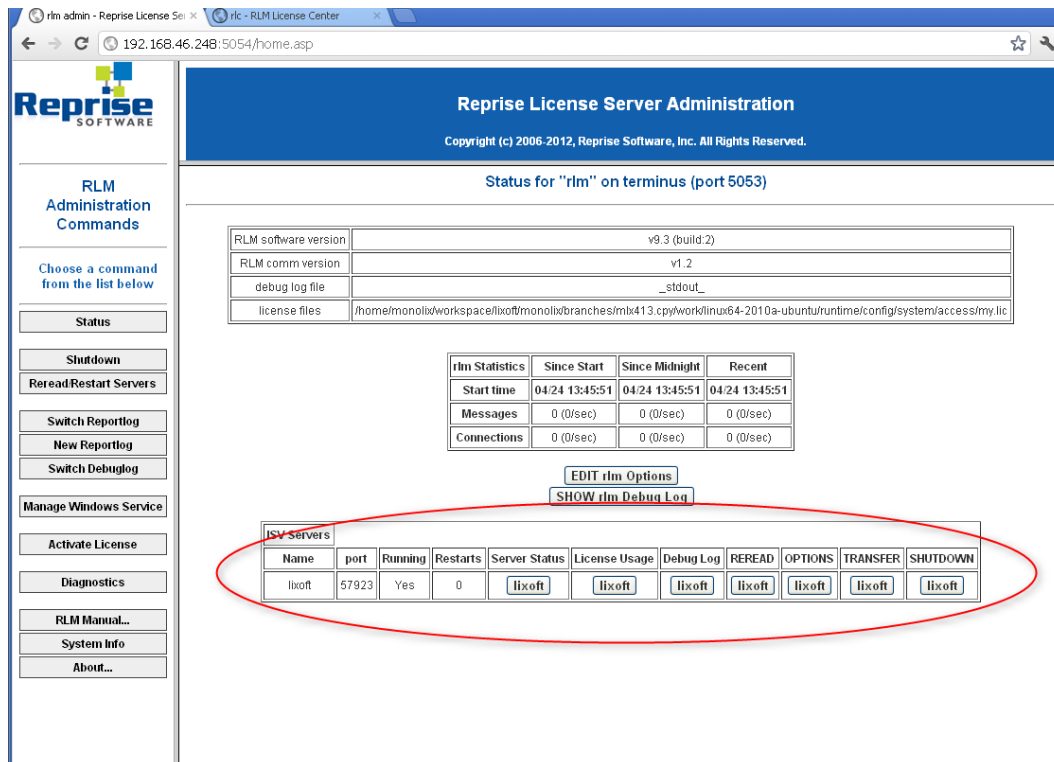
- The RLM server is executed on the computer `master-server`.
- Monolix is installed on each cluster node of the cluster.
- The license file is copied on **each cluster node** in the directory `<monolix install path>/config/system/access/` for the Matlab version of Monolix or `<monolix install path>/bin/Monolix_mcr/runtime/config/system/access` for the standalone version.
- Each monolix user runs Monolix from the cluster node.
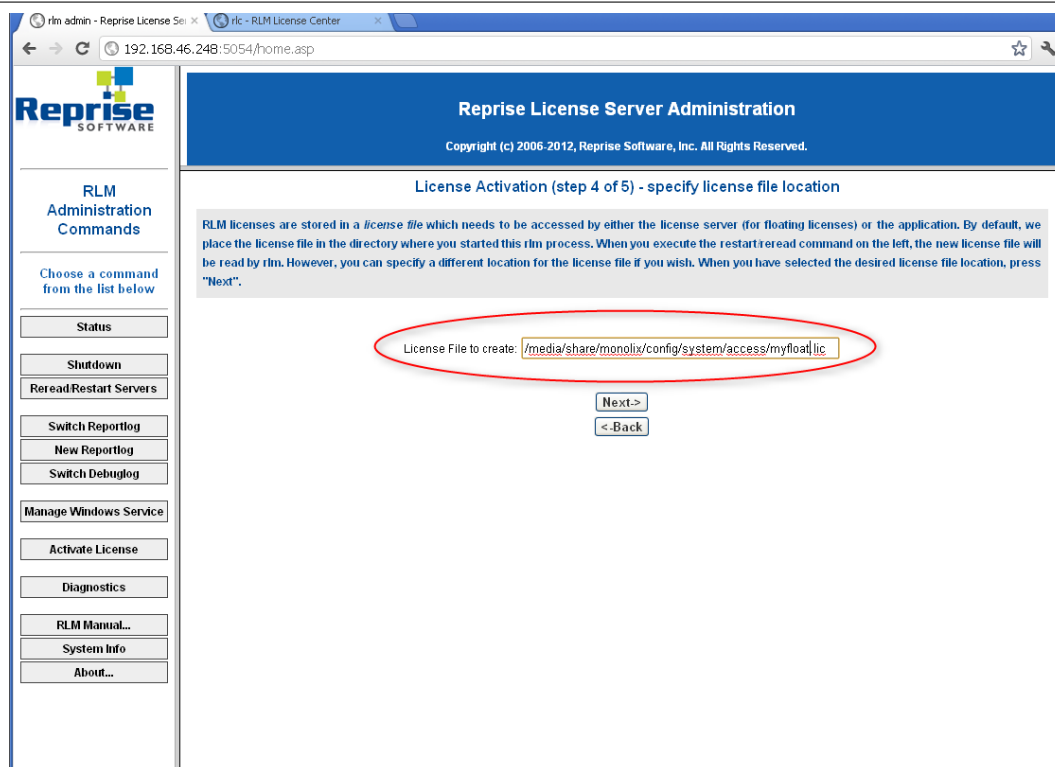
7. Stop the server manually and restart it from the directory (or use option `-c`)

- `<monolix install path>/config/system/access/` for the MATLAB version
- `<monolix install path>/bin/Monolix_mcr/runtime/config/system/access` for the standalone version of MONOLIX .

Now RLM is running with the provided license. This is verified in the web interface by clicking on `status` button.

8. RLM Server : server hostname and port considerations.
   If for any reason, the server port or the server hostname is not registered in a DNS, it is possible to change these informations directly on licence file.
   The line `HOST <hostname> <mac> <port>` can be changed by `HOST <rlm server ip> <mac> <new port>`.

9. RLM Server : firewall considerations.
   If the RLM server is behind a firewall, the port 5053, 5054 and the ISV port have to be opened.
   The ISV port can be set directly in license file by changing the ISV line as follow:

   ```
   ...
   ISV lixoft port=<your ISV port>
   ...
   ```

10. Managing RLM server :
    The documentation of the management of the RLM server provided by Reprise Software is available at
    `http://www.reprisesoftware.com/RLM_Enduser.html`

Roaming license

RLM has the ability to allow a floating license to roam to a system which will subsequently be disconnected from the network for a short period of time. The resulting license can be used for the number of days specified when the license was set to roam, and is checked back in automatically at the end of this. In addition the user can return the roamed license back to license pool early if this is desired.

See `License activate tools` (which can be launched from the MONOLIX interface, in `tools` menu)
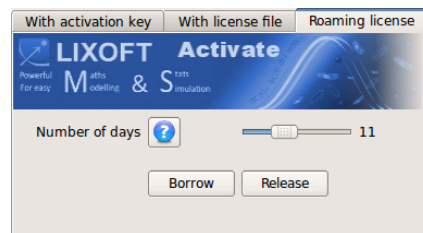
This feature is enabled on demand. An extra activation key will be provided by LIXOFT and the procedure to get the roaming license feature is identical to the installation of a floating license. To enable this feature, the file `system.xmlx` (stored in directory `<monolix install path>/config/` -MATLAB version- or `<monolix install path>/bin/Monolix_mcr/runtime/config/` -standalone version of MONOLIX - must be modified by setting to `"on"` the roaming option:

```
<?xml version="1.0" encoding="utf-8"?>
<monolix>
    <preference>
        <session>
            <userPath windows="%USERPROFILE%" linux="$HOME"/>
            <license activation="http://activate.lixoft.net" roaming="on"/>
        </session>
    </preference>
</monolix>
```

2.3 ChangeLog

```
1  Monolix 4.2.2 (2013-03-15)
2  Bugs Fixes:
3      - MLXTran Project
4          - fixed omega and error model parameters was not correctly saved
               when they were set as 'Fixed'
```

```
 5        - load data file error when the data path was not ended by a '/'
 6        - under Windows OS, when the result folder is not the project
            folder, the path was prefixed by a '/'
 7        - parser did not load correlation block
 8
 9     - Graphics : Spaghetti plot
10        - error when there are censored data on several outputs (index out
            of bounds)
11
12     - MLXTRAN model :
13        - a zero-order dose absorption (or infusion) combined with a lag-
            time could raise an error of bad dimension if the end of the
            absorption followed the lagged dose before the next record
14        - the autocorrelation was not correctly set when the error model
            is defined inside the structural model
15
16     - GUI
17        - when the error model is defined in the structural model and  it
            is changed from constant or proportional to combined, some
            parameters became fixed to 0
18        - problem handling IOV + categorical covariates dependency on
            omegas
19        - problem with Extensive workflow
20        - problem with "View the Data" when the number of outputs in the
            dataset is different than the number of outputs in the
            structural model
21        - issues handling covariates with only one category
22        - error when loading project when there were results computed with
             version prior to 4.2.0 with censored data
23        - Hypothesis test on residual error interface, allows only to
            change continuous error models that are not defined in the
            structural model
24
25     - Algorithms:
26        - conditional mode estimation did not worked for IOV  when there
            are not variaibility at all
27        - projects with autocorrelation and censored data crashed
28        - log-likelihood estimation crashed when using band error model
             and the structural model returned values outside the band
29        - Axe titles were not correctly shown with correlation and
            discrete outputs
30        - inter-occasion variability
31          - issues when there was no variability at the first or at the
              last occasion level
32          - issues when using correlations in iov
33        - Mixtures
34          - BSMM + autocorrelation failed
35          - BSMM with several outputs failed
```

```
36              - issues when using several latent covariates
37              - conditional expectation estimation failed when using several
                  MC chains
38          - Bayesian analysis failed if there were covariates with more than
              2 categories
39
40      - Monolix user API
41          - Tables with individual contribution to log-likelihood (
              indContTable) and covariates information summary (covTable) did
              not work
42
43  Enhancements:
44      - GUI
45          - Errors due to problems loading results does not stop the load of
              project
46
47      - Algorithms
48          - Covariates with only one value or category are not accepted
              anymore for population parameter estimation
49
50      - Monolix user API
51          - documentation updated
52          - Arguments 'Toggle' changed to 'Workflow' to reduce ambiguity
53          - Added new function to know if Monolix packages is compatible
              with Matlab version
54
55  --------------
56
57  Monolix 4.2.1 (2013-02-15)
58  Bugs Fixes:
59      - MLXTRAN Project : in STRUCTURAL_MODEL section resolved problem
              with path relative to %MLXPROJECT%
60      - mlxEditor, mlxPerlScript : under Suse Linux OS, conflict with
              libstdc++ and Qt librairies installed on the OS.
61      - Graphics : Kaplan Meier
62        - mean normalization
63        - survival curve: case of censured data
64      - simulations where wrong in presence of correlation between
              individual parameters
65      - MLXTRAN Model :
66        - Events could be close at a numerical epsilon for the solver, but
                not for the solver driver
67          Rarely, it resulted into an explicit integration failure,
                returning "NaN"
68        - For the simulation of RTTE models, the ordering of the output
              names had to be alphabetical
69        - Not declaring all regression variables that where selected from
              the data set crashed the application.
```

```
70          - Declaring some PK without actual doses within the data set
               raised an error.
71          - Using the deprecated syntax with several lagged compartments
               returned "NaN"
72       - Algorithms
73          - Error when some subjects had no doses in conditional mode
               computation
74       - GUI
75          - "Display the data" button did not update the information when
               the dataset was changed after running algorithms
76          - Convergence assessment GUI failed when there where only one
               individual parameter
77          - structural models with several dots (.) were not compiled when
               clicking in the compile button in the Model selection GUI
78          - projects with more outputs in structural model than observations
               in dataset caused an error when it is loaded
79          - the editor was not saved in the preference file
80       - Convergence assessment graphics did not handled correctly when
            there were not variances on some parameters or their covariate
            dependence
81
82    Enhancements:
83       - add possibility to configure the compiler (used to create
            Structural Model plugins) through the file 'system.xmlx'
84       - user API:
85          - it is possible now to use matlab function "ver" to know Monolix
               version and Monolix API version
86       - mlxEditor:
87          - allow multiple files selection on open file dialog box
88          - add 'Find and replace'
89          - set tabs movable
90       - MLXTRAN Model :
91          - Continuous observations can be declared within the model.
92          - Macro for a depot absorption, with a target ODE component.
93       - Permutation kernel for mcmc included
94
95    Other:
96       - Licensing system : '.ini' files desactivated (only the '.lic'
            files are allowed)
97       - residual error models in main interface are shown now with their
            full name (those used in MLXTRAN project and model)
98    --------------
99
100      Monolix 4.2.0 (2012-11-26)
101
102   Bugs Fixes:
103      - MLXTRAN Project : in OBSERVATION section when a prediction has the
            same name as an individual parameter the project parses fail
```

```
104        - PerlScript : bug with parameter '--use-matlab=false' was taken as
             'true'
105        - Identity line works in observations vs predictions graphic
106        - Prediction distribution : percentiles are correctly displayed
107        - Color when stratify in covariables graphic
108        - Problem with prior (by default prior is Variance and not Standard
             Deviation, this implies a syntax error (standardDeviation <->
             variance)
109        - Wrong data file for the demonstration project
             rtteWeibullCount_project.mlxtran
110        - "Display the data" button did not work
111        - bug when unchecking and checking "random effects" variability in
             simulation interface
112
113  Enhancements:
114        - Interval censoring for continous data
115        - Extended priors on fixed effects
116        - Mlxtran model and Mlxtran project editor
117        - Perl script HMI
118        - Autosave
119        - Multiple covariate definitions
120        - Add batch-mode demo
121        - Add a doc package and a rlm server package (floating license
             server)
122        - Graphic
123           - BLQ graphic : possibility to choose his own interval of
                censored data
124           - Reorganisation of panel for list of graphics
125           - Background color for each graphic in preferences
126           - When split, limits are the same  for all axes
127           - Obs. vs Pred., observations can be relied by individual
128           - Optimal bandwidth setting for parameter distribution
129           - CvSaem graphic : choice of axes number
130        - Interval-censored data and maximum number of events for time-to-
             event and drop-out data models
131        - Markov chain for categorical data
132        - Continuous-time Markov process for categorical data
133        - probit and normal cdf for Mlxtran model
134        - New user API including simulation-estimation, convergence
             assessment and simulations tools
135        - Possibility to define new covariates as transformation of already
             defined ones
136
137  New graphics:
138        - Posterior and prior functions for bayesian
139        - Individual contribution for the LL
140        - Transition probabilities
141        - Kaplan-Meier survival function
```

```
142
143  New tables:
144      - Individual contribution to log-likelihood
145      - Covariates summary
146  --------------
147
148    Monolix 4.1.4 (2012-07-16)
149
150  Bugs Fixes:
151      - Saving preferences from tools menu failed.
152      - Display remaining time (license) correctly.
153      - Problem with license activation file path.
154      - Add license agreement into Linux installer.
155      - The horizontal slider in "Check initial fixed effects" interface
            did not appear for some number of individual parameters.
156
157  Enhancements:
158      - Windows 64 RC.
159      - Management of the maximum number of threads for MLXTran models (
            can be set from the preference tools: MonolixGUI ->Tools ->
            Preference)
160      - License activate: inform user to not set activation key
161        if the license is a floating license.
162      - Documentation :
163          - Installation guide : Windows 64 bits.
164          - User Guide : Cluster section revised.
165          - Model MLxTRAN : list of keywords of the language.
166
167
168  --------------
169
170    Monolix 4.1.3-sp2 (2012-05-29)
171
172  Enhancements:
173      - system.xmlx : possibility to not display Lixoft Activate.
174      - Lixoft Activate : add the possibility to send an email with
            encoded computer information to create license @Lixoft.
175      - Lixoft Activate : manage "cannot connect to url" error by asking
            user to go on a web site or send an email.
176
177  Bugs Fixes:
178      - IOV Problem with R2010bSP1
179      - perlScripts : bug in the management of the configuration file for
            [program-execute-options] and run on a cluster.
180      - add 'rlmutil.exe' for windows packages (forgotten in previous
            packages).
181      - problem floating license.
182      - warnings for occasions without dose were removed.
```

```
183        - when the last Individual/Occasion had no dose , Monolix crashed .
184        - When there were syntax errors in the structural model , monolix
              said that it could not find the file instead giving the MLXTRAN
              message
185        - NaN observations are now mentionned as error when algorithms are
              launched .
186        - Update documentation : in batch mode section , there is a bad path .
187  --------------
188
189    Monolix 4.1.3 - sp1 (2012 -05 -21)
190
191  Bug Fixes :
192      - GUI :
193          - Check Initial Fixed Effects interface crashed when creating
                covariate and parameter 's sliders for some sizes
194  --------------
195
196    Monolix 4.1.3 (2012 -05 -02)
197
198  New Features :
199      - MLXTran model : allows negative categories
200      - License management : uses RLM as license provider
201      - Compiler manager : adds the possibility to choose the embedded
            compiler
202      - The Monolix and Matlab versions are now stored in the algorithm
            result files
203
204  Bug Fixes :
205      - MLXTRAN project :
206          - continuous transformation can take a mathematical expression
207          - problem with structural model path
208      - MLXTRAN model :
209          - Under Linux 64 bits , due to library conflicts with Matlab
                R2010b and better , the multi - threaded
210            loading of the model description for the project occasionally
                  fails
211          - Only the last table variable is recorded , overwriting the
                first one
212      - Graphics :
213          - log / linear works on all graphics
214          - when log - log scale is set for " observed versus predicted ', the
                diagonal line isn't displayed anymore
215      - GUI :
216          - editor call did not work
217      - Algorithms :
218          - bug for individuals without some type of observations and with
                IOV computing conditional mode
219          - bug when there were continuous outputs after discrete outputs
```

```
220          - Fisher Information Matrix by Stochastic Appoximation does now
                 handle better the case when there are
221            no parameters to estimate in the residual error
222      - Session:
223          - when the directory monolixData/monolix<version> is renamed
                 during an active Monolix session, stopping
224          - Monolix caused an exit of Matlab.
225

226  --------------
227
228    Monolix 4.1.2: (2012-03-05)
229
230  --------------
231
232  New Features:
233      - PerlScripts : possibility to save the results in the project
             directory instead of the output directory
234      - In system.xmlx : automatically creates a directory hierarchy for '
             monolixData' path
235
236  Enhancements:
237      - MLXTran (structural model) multi-threading processing enhancement
238
239  Bug Fixes:
240      - Batch Processing failed when a very large number of projects were
             launched
241      - MDV column: when MDV=2 only the regression variables were taken
             into account
242      - Fixed a bug in graphics saving
243      - Fixed error when an empty result folder was timestamped
244      - Simulation of categorical data, whenever no category 0 is defined
245      - Fixed take into account UserPath defined in 'system.xmlx' for the
             preference file saving
246

247
248  --------------
249
250    Monolix 4.1.1: (2012-02-13)
251
252  --------------
253
254  New Features:
255      - timestamped backup
256      - preferences interface
257      - tools menu for activating license and preferences
258      - option for locking structural model modifications
259      - Project-MLXTRAN grammar modification : initialization of parameter
             is written now as beta_{pi,cov}, pop_{pi}, omega_{pi}, ...
```

```
260     - save graphics as png / ps / jpg / bmp or tiff
261     - selection of graphics/tables to be saved
262
263 Bug Fixes:
264     - Project-MLXTran: user can define the result folder
265     - LoQ difference between 3.2 and 4.1
266     - statistical test for error model and covariate model
267     - xmlx loading from 3.2 to 4.1
268     - correlation (levelName consistence with IOV) + parser error
269     - observation model (prediction = observation name)
270     - path for MONOLIX user profile can include special characters
271
272
273
274 --------------
275
276    Monolix 4.1.0: (2012-01-23)
277
278 --------------
279
280 psmlx:
281     - compatible with the mlxtran format of projects
282     - available on Windows OS
283
284 mlxtran:
285     - new syntax
286     - PK macros
287     - RTTE models
288
289 license:
290     - interface for installing the license file
291
292 Interface:
293     - setting for axes' limits
294     - information for the observation model
295     - shortcut for model libraries
296
297 File system:
298     - improved handling of special characters for filepaths
299
300 Demos:
301     - updated for the new mlxtran syntax
302     - dispatch of the model library for demos
303
304 Known Bugs:
305     - under Windows OS, user directory cannot contain special characters
            other than spaces
306
```

```
307
308
309   - - - - - - - - - - - - -
310
311     Monolix 4.0.1: (2011 -10 -27)
312
313   - - - - - - - - - - - - -
314
315   psmlx:
316       - use-matlab option didn't work in command line mode
317       - multi-threading : multhreading didn't work
318       - take account the p.coded files
319
320   mlxtran
321       - problem with FIM options : both linearization and
              stochasticApproximation appeared after a save with
              stochasticApproximation option set
322       - avoid the unloading of project when settings files does not exist
              : default settings are loaded
323   license:
324       - multi write database didn't work well in multi-threading mode
325
326   Interface:
327       - save lists
328       - configuration panel
329       - launching some graphics alone is now possible
330       - the graphics were closed when "Use last estimates" were used
331       - when monolix was launched twice without loading or creating a
              project, two toolbars were created
332
333   Algorithm and simulations:
334       - simulation works now with dataset without EVID column and with MDV
              column
335
336   Results:
337       - the graphics fit now to the paper in .ps files
338       - xLabels were wrong for some graphics when several regression
              variables were present
339       - some graphics  crashed when launched after some hypotheses tests
              were done
340       - Visual studio redistribuable problem
```

2.4 Troubleshooting

2.4.1 Downloading Monolix

Problem: *My web browser claims that the* Monolix *download website has insufficient reputation and suggests to stop the download.*

Solution: Some browsers like *Google Chrome* and *Internet Explorer* may ask whether to keep or remove the Monolix archive just after download because of the insufficient reputation of the Monolix download website, simply because it is not referenced, as opposed to the Lixoft website. Please ignore the warning and choose to keep the file. You can use a MD5 tool to verify that the downloaded file is not corrupted.

Problem: *The* Monolix *archive is removed just after being downloaded.*

Solution: Some antivirus may consider the Monolix archive as risky and put it in *quarantine* or remove it. This is due to the fact that Monolix embeds a compiler for the Mlxtran language. Two solutions are available:

1. Deactivate your antivirus auto-protection process during download and installation, or

2. Restore the file from the quarantine.

To restore the file from quarantine, please refer to the documentation of your antivirus software. For the most common examples:

- *Norton Antivirus 2012:*

  - Start *Norton Antivirus*
  - Choose `Advanced`, then `Quarantine`

- *Avast Antivirus 7:*

  - Open *Avast*
  - Choose `Maintenance`, then `Virus Chest`

You should see the downloaded file among the quarantined files. Execute the `Restore` action; the archive will be restored into the directory used for downloading. Click on the archive (ignore a possible "malware" warning, again related to the fact that Monolix embers a compiler.), and installation will start.

2.4.2 Running Monolix

Problem: *When launching the standalone version, my antivirus tells me that the file* `mlxinitializer.exe` *is risky.*

Solution: If your antivirus apparently removed the file `mlxinitializer.exe`, check if it was actually put on *Quarantine*, or removed. If it is in *Quarantine*, please restore it by following the same instructions as provided above. If the file was removed you will need to reinstall Monolix .

You should be able to add this file to your antivirus *Trusted Zone* or *Trusted files*.

- *Norton Antivirus 2012:*

    - go to folder `Monolix/monolix422s/bin` in installation directory: for instance `c:/ProgramData/Monolix/monolix422s/bin`

    - right click on `mlxinitializer.exe`, click on `Norton Antivirus`, then `Norton File Insight` then look for 'Unproven', and click 'Trust Now'.

- *Avast 7:* This software may start Monolix in a *SandBox*, i.e in a zone where the antivirus avoids any modification of the system or the files. He will ask you what to do at each run. Select *Run normally.*

    You can also add `mlxinitializer.exe` to the exclusions in its *Auto-Sandbox* settings: option `Additional Protection/AutoSandbox` and then click on `Settings` button.

Chapter 3

Using Monolix

3.1 Introduction

In order to use Monolix, your problem must be described as a Monolix project. A project specifies

- the dataset to use

- the structural model

- the statistical model, which includes

  - covariate model for individual parameters

  - covariance model for random effects

  - observation model

- the tasks to run, their settings and initial values

The dataset is an `ASCII` file containing all the data for your study (see Section 3.3.1 for a brief description) and the structural model is a function explaining the theoretical model behind your data. It can be easily described using the Mlxtran language (see the document `modelMLXTRANtutorial.pdf` in Monolix documentation folder), or as a Matlab function for a Matlab version of Monolix.

A mathematical description of the statistical models handled by the software can be found in Appendix A.

All these informations can be set in the MONOLIX main window.

In this chapter, we will describe the main window and its different sections, using a simple example.

3.1.1   The theophylline example

We will consider the theophylline data (see [6, 21]) as an example to illustrate how to use MONOLIX.

In this case subject i receives a dose D_i per kilo at time 0 and serum concentrations (y_{ij}) are measured at times (t_{ij}). Serum concentration is modeled by a first-order one compartment model. Then,

$$y_{ij} = \frac{D_i\, ka_i}{V_i ka_i - Cl_i}\left(e^{-\frac{Cl_i}{V_i} t_{ij}} - e^{-ka_i\, t_{ij}}\right) + a\varepsilon_{ij} \tag{3.1}$$

where ka_i is the absorption rate constants of subject i, V_i is the volume per kilo of subject i and Cl_i is the clearance per kilo of subject i. These three parameters are nonnegative real numbers and are assumed to be log-normally distributed. Here,

- the vector of regression (or design) variables is $x_{ij} = (D_i, t_{ij})$.
- the vector of individual parameters is $\psi_i = (ka_i, V_i, Cl_i)$,
- the model is assumed to be homoscedastic and $g \equiv a$.

The only available covariate is the weight (w_i) of the subjects.

The data file `theophylline_data.txt` is in the directory `<demos folder>/theophylline`. This file is in the so-called "NONMEM format" and contains the ID numbers of the subjects, the doses per kilo (D_i/w_i), the times (t_{ij}), the observed concentrations (y_{ij}), the weights (w_i) and the genders (s_i) (this additional covariate is not in the original dataset: it was added for the demo).

ID	AMT	T	DV	WEIGHT	SEX
1	4.02	0	.	79.6	M
1	.	0.25	2.84	79.6	M
1	.	0.57	6.57	79.6	M
⋮	⋮	⋮	⋮	⋮	
2	4.4	0	.	72.4	M
2	.	0.27	1.72	72.4	M
⋮	⋮	⋮	⋮	⋮	
12	5.3	24.15	1.17	60.5	F

Here, "AMT" (amount per kilo) holds for "D/w", "T" is the time and "DV" (dependant variable) holds for "y".

3.2 The main window

After starting MONOLIX, the main window appears empty. Only the "`Project`" and "`?`" (help) menus are available, there you need to choose or create a project to continue.

The project can be created from scratch by clicking on `Project->New` in menu or modifying a previously loaded one (`Project->Load`).

It is then possible to save it with the menu `Project->Save` in any of three formats: as a MLXTRAN project (`ASCII` file with extension `.mlxtran`, recommended and default for new projects), as a binary MATLAB (`.mat`) file or as a XML (`.xmlx`) file.
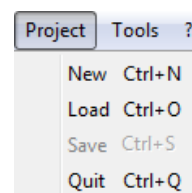
Once the project is chosen, the different sections of the interface become visible. Those sections facilitate the project definition.

All the menu options also appear and the toolbar buttons become enabled.

- create a new project

- load an existing project

- save the current project (`mlxtran`, `mat`, and/or `XML` file)

- see the last results for the current project

- display the data

- estimate the population parameters,

- estimate the Fisher information Matrix and the standard errors,

- estimate the individual parameters,

- estimate the log-likelihood,

- display a set of graphics

- test the covariate model

- test the covariance model

- test the residual error model

- simulate a new dataset

- publish the results (not available with the standalone version)

- about MONOLIX

- quit MONOLIX

In our example, the project file `theophylline_project.mlxtran` is included in the `Demos` so you can load it or create a new one from scratch.

- To load the project, use the button and select `theophylline_project` in the `<demos folder>/theophylline` folder (go to Section 3.7 to see how to run the algorithms).

- To create a new project to analyze the theophylline data, use the button and follow the instructions below.

3.3 The "Data and Model" frame

3.3.1 The data

The data file should contain a matrix in an `ASCII` format with or without header for each column. The columns of this matrix contain (in any order)

1. the ID of the subjects (can be any string or number, not necessarily ordered),

2. the regression variables (x_{ij}),

3. the observations (y_{ij}),

4. the covariates,

5. additional information (censoring, rate, tau ...).

▷ *theophylline example:*

To select the theophylline dataset use the button $\boxed{\texttt{The data}}$. A new window is opened
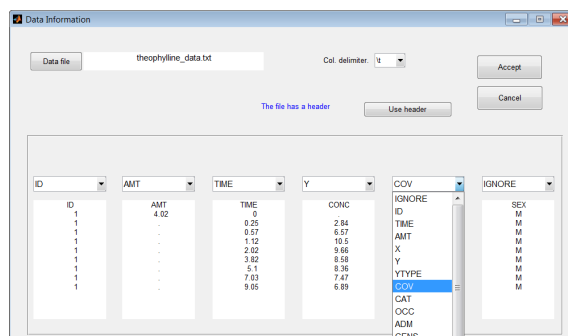

- Select the data file `theophylline_data.txt`. The dialog box to select the dataset can be opened with the button $\boxed{\texttt{Data file}}$.

- If MONOLIX does not recognize the different columns you can choose one of the column delimiters from comma (","), semicolon (";") , space (" "), tab ("\t"):
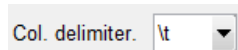
- Since the file has a header in the first line (`ID AMT T DV WEIGHT SEX`), MONOLIX will use the recognized columns by default. You can choose other headers for each column by yourself. Use $\boxed{\texttt{Use header}}$ button whenever it is desired that MONOLIX uses the header from the file, instead of the current one.

- Check that MONOLIX has recognized the keywords (`ID, AMT, T, DV`) and translated them to (`ID, AMT, TIME, Y`).

- The headers `WEIGHT` and `SEX` are not recognized and set to `IGNORE`. Nevertheless, you can consider the weight as a **continuous covariate** by selecting the label `COV` for this column and the gender as a **categorical covariate** by selecting the label `CAT` for this column.

- Accept the data information with the $\boxed{\texttt{Accept}}$ button

A list of the main headers identified by MONOLIX is

- `ID` (or `#ID, I`) to identify subjects

- `TIME` (or `T`) for the time

- `AMT` (or `DOSE, D`) for the dose

- `X` (or `REG, XX`) for any regression variable

- `Y` (or `DV, CONC`) for the observations

- `YTYPE` (or `ITYPE, TYPE, DVID`) for the type of observations when there are several types of observations (1 for the first type, 2 for the second type, ...). `YTYPE` is not necessary in the case of a single output.

- `COV` for the continuous covariates

- `CAT` for the categorical covariates

- `OCCASION` (or `OCC`) for the occasion

- `ADM` for the type of administration

- `CENS` for censored data. Can be -1 to represent right-censored data

- `LIMIT` for interval-censored data. It gives the lower limit while `Y` gives the upper limit

- `RATE` (or `R`) for the infusion rate

- `TINF` for the infusion duration

- `SS` for steady-state (requires column `II`)

- `ADDL` for the number of additional doses (requires column `II`)

- `II` (or `TAU`) for the inter-dose interval

- `MDV` (Missing Dependent Variable) `MDV= 0` if the row contains an observation and `MDV=1` otherwise. `MDV` is not necessary if a dot is used to say when a row does not contain any measurement (`Y='·'`). You can use `MDV=2` to include times for regression variables updates and for prediction evaluation (see Section 4.11).

MONOLIX 4.2.2
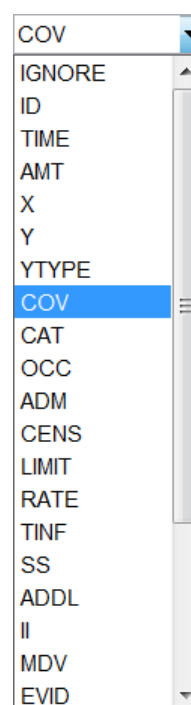
- EVID for Dose events. `EVID` is not necessary if `DOSE`='·' is used when a row does not contain any dose information. `EVID=4` resets the system to the initial state.

3.3.2 The model function

Use `The model` button to define the model function. In the `Model List` window, it is possible to select a model from the model library included in the software (PK library, PD library, VK library, discrete data library) by using the `Monolix Library` button; or from a personal list by using the `Other list` button.
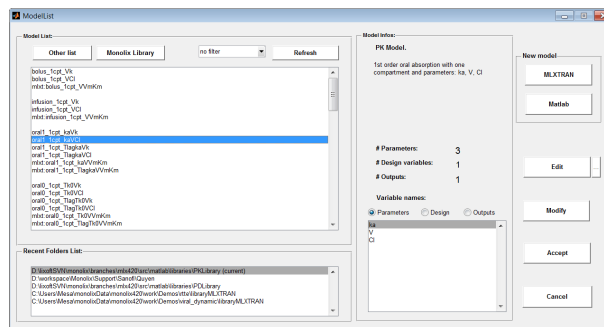
Refer to `modelMLXTRANtutorial.pdf` for more details of how to implement your own model using MLXTRAN.

When a model is selected from one of these lists, the model information is summed up in the `Model Info` window: name of the function, number and names of the parameters, of design variables and of outputs.

Edit the model with the `Edit` button. If you use the MATLAB version of MONOLIX, then the MATLAB editor will be used by default. If you use the standalone version of MONOLIX (or if you want to use another editor with the MATLAB version), select your own editor with the `···` button. In standalone version, only the MLXTRAN models can be edited, and the only `.m` models that can be used are the built-in models.
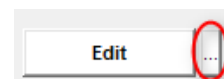
Note: MONOLIX includes now its own editor for MLXTRAN projects and models. Set the editor to 'mlxEditor' if you want to use it.

▷ *theophylline example:*

We use the first order oral absorption with one compartment model function (`oral1_1cpt_kaVCl`) from the PK library which has 1 design variable (time), 3 parameters (k_a, V, Cl) and 1 output (concentration).

Important: When a project is created (or loaded), there are two ways to change the structural

model:

- by clicking on the button `The structural model`.

- by right-clicking on the name of the model.
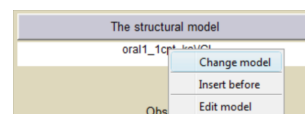
3.3.3 The covariate model

Here, m is the number of covariates and d is the number of individual parameters. Then, `The covariate model` is a $m \times d$ matrix A containing only 0 and 1. For any $1 \leq k \leq m$ and any $1 \leq \ell \leq d$, $A_{k,\ell} = 1$ if the ℓ-component $\phi_{i,\ell}$ of the individual parameter ϕ_i is function of the kth covariate, and 0 otherwise.

▷ *theophylline example:*

Let us assume that no covariates are used in the model:

$$
\begin{aligned}
\log(ka_i) &= \mu_1 + \eta_{ka,i} \\
\log(V_i) &= \mu_2 + \eta_{V,i}, \\
\log(Cl_i) &= \mu_3 + \eta_{Cl,i}
\end{aligned}
$$

Consider now that the log-volume is a linear function of the weight and that the log-clearance depends on the gender.

$$
\begin{aligned}
\log(ka_i) &= \mu_1 + \eta_{ka,i} \\
\log(V_i) &= \mu_2 + \beta_{W,V} w_i + \eta_{V,i}, \\
\log(Cl_i) &= \mu_3 + \beta_{S,Cl} \mathbb{1}_{s_i=M} + \eta_{Cl,i},
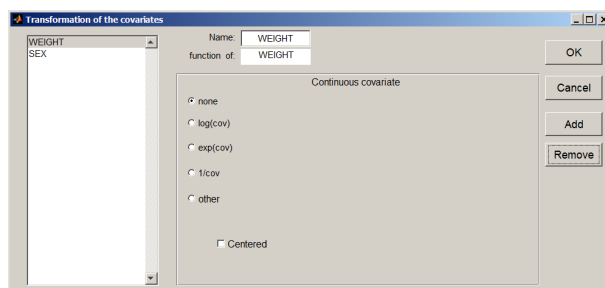\end{aligned}
$$

Here, the reference group are the female and the population parameters are

$$
\begin{aligned}
\log(Cl_{\text{pop,F}}) &= \mu_3 \\
\log(Cl_{\text{pop,M}}) &= \mu_3 + \beta_{S,Cl}.
\end{aligned}
$$

3.3.4 Creating and transforming covariates

Some times, it is needed to use some transformed version of the original covariates. It can be done in the window opened when clicking on the button `transform`:

1.- <u>transforming continuous covariates</u>

It is possible to consider non-linear functions of the continuous covariates. Logarithmic, exponential or fractional polynomials are available and personal functions can be considered. This is also possible to center the covariates.
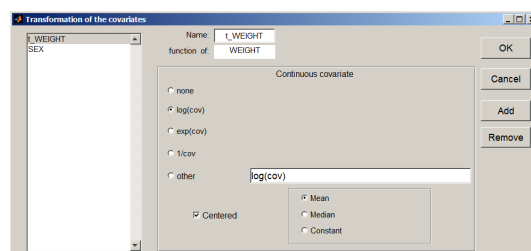
▷ *theophylline example:*

Consider for example, a log-transformation of the weight, centered by the mean:

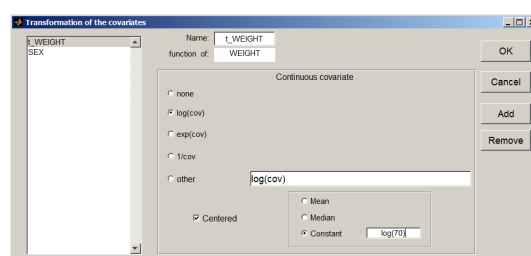$$w_i^\star = \log(w_i) - \log(\overline{w})$$

Then, the covariate model used for the volume is

$$V_i = V \left(\frac{w_i}{\overline{w}}\right)^{\beta_{W,V}} e^{\eta_{V,i}}$$


Instead of using the mean to center the log-transformation of weights, you can normalize the weight by some constant value a ($a = 70$ for example) by centering the transformed weight by $\log(70)$:

$$V_i = V \left(\frac{w_i}{70}\right)^{\beta_{W,V}} e^{\eta_{V,i}}$$

Remark: The pre-defined transformations will be removed in a future version of MONOLIX. We strongly recommend to write your own transformation using the "other" option. For example, the allometric transformation of the weight proposed above just reduces to:
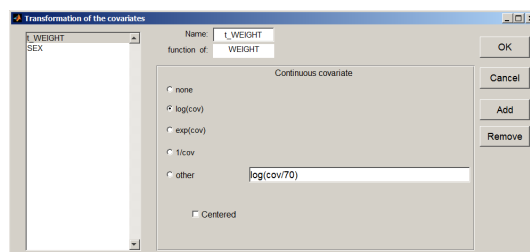
2.- transforming categorical covariates

You can easily modify the groups defined by a categorical covariate. You can also select the reference group and modify the names of the groups (just click on G_k to change the name of the k-th group).
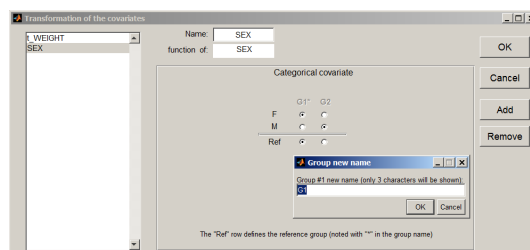
3.- creating new covariates

In some cases, it is needed to use different transforms of the same covariate, for instance to use `WEIGHT` for some parameter and log(`WEIGHT`) for other. It is possible by creating a new dependent covariates.

To create new covariates, click on `Add` button and select from which covariate, among the originals and already created ones, the new one will depend. Then click on `OK`.
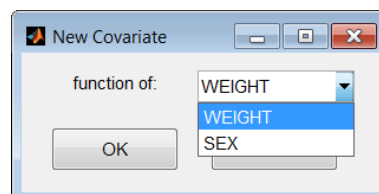
The new covariate is added to the list at the left so you can use them like the original ones and define their transformations.

The new or transformed covariates receive a name by default that can be modified by the user at the top of the window.
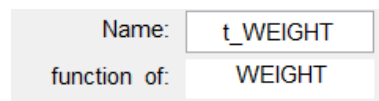
It is also possible to remove the new covariates or the transformations from the original covariates with the button `Remove`.

It is important to note that, the new covariates will be of the same type (continuous or categorical) than those from which they are function of, and that new categorical covariates can only be function of original (not transformed) categorical covariates.

Note: It is recommended to set the name of the modified or newly created covariates before creating new ones.

3.3.5 Distribution of the individual parameters

The default distribution of the individual parameters are defined by the structural model, but it is possible to change it in the MONOLIX window by clicking on the buttons below `Distribution of the individual parameters` to
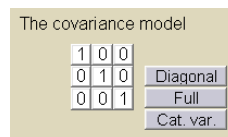
- `N` for a normal distribution $\psi = \varphi$

- `L` for a log-normal distribution
  $\psi = e^{\varphi}$

- `G` for a logit-normal distribution
  $\psi = (1 + e^{-\varphi})^{-1}$

- `P` for a probit-normal distribution
  $\psi = \mathbb{P}(\mathcal{N}(0,1) \leq \varphi)$

- `B` for a power-normal (Box&Cox) distribution
  $\psi = (A\varphi + 1)^{\frac{1}{A}}$

- `C` for a custom (user defined) distribution
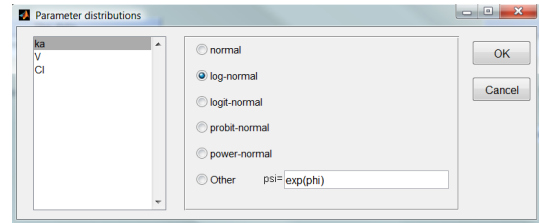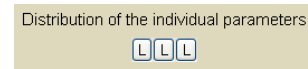
3.3.6 The covariance model of the random effects

The `covariance model of the random effect` is a $d \times d$ matrix Δ formed by 0 and 1. For any $1 \leq j \leq d$ and any $1 \leq l \leq d$, $\delta_{jl} = 1$ if there is a correlation between η_{ij} and η_{il}, and $\delta_{jl} = 0$ elsewhere. $\delta_{jj} = 0$ means that the variance of the jth random effect is 0.

A diagonal covariance matrix is obtained by setting the matrix Δ to

Assume now that the absorption rate does not vary between subjects and that clearance and volume are correlated. Then, Δ should be set to

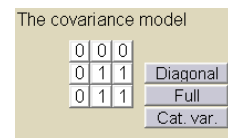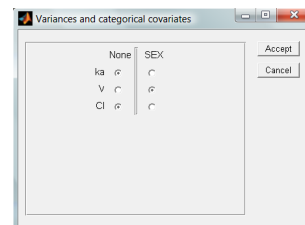**Note:** The covariance matrix, must be symmetric and block defined. For this reason, if the matrix is not valid the algorithms can not be launched. Also, starting from MONOLIX 4.2.2, the project file can not be saved when the covariance matrix is not valid.

You can also assume different variances in the groups defined by a categorical covariate (click on the button $\boxed{\texttt{Cat.var.}}$).

Assume for example that the variance of the volume depends on the gender
Remark: In the different groups, the random effects have different variances but the same correlations. In this example, the correlation between Volume and Clearance is the same for male and female.
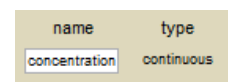
3.3.7 The observations model

`The observations model` can be defined in the structural model function (for discrete data: count, categorical, RTTE, etc) or it can be a residual error model (for continuous data).

The observation model section shows, for each observation, the variables name and their type following the structural model definition. If the observation name is not set in the structural model (for continuous observations, it can be given the prediction name instead, see MLXTRAN models description) then it can be defined by the user using the interface.

For the continuous observations, we consider the general form $y = f + g\,e$ where e is a sequence of independent random variables normally distributed with mean 0 and variance 1.

Some extensions assume that there is a transformation t such that $t(y) = t(f) + g\,e$. It is also possible to assume that the residual errors are correlated. Here are some examples of error models:

const: constant error model $y = f + a\,e$
prop: proportional error model $y = f + b\,f\,e$
comb1:   combined error model $y = f + (a + b\,f)e$
comb2:   combined error model $y = f + a\,e_1 + b\,f\,e_2$
propc: proportional error model + power $y = f + b\,f^c\,e$
comb1c: combined error model + power $y = f + (a + b\,f^c)e$
comb2c: combined error model + power $y = f + a\,e_1 + b\,f^c\,e_2$
exp: exponential error model $t(y) = \log(y)$ and $y = f\,e^{a\,e}$
logit: logit error model $t(y) = \log(y/(1-y))$
band(A,B):   extended logit error model $t(y) = \log((y-A)/(B-y))$
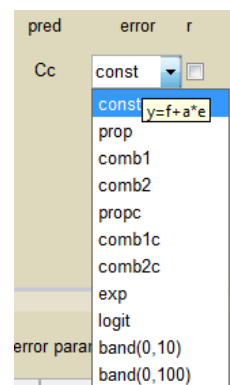
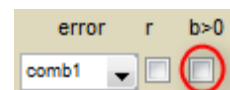Select the checkbox `r` if you want to consider autocorrelation on the residual errors.

You can also constrain the second parameter `b` in `comb1` and `comb1c` error models to be positive by checking `b>0`

3.4   The "Initialization" frame

Initial values are specified for the `Fixed effects`, for the `Variances of the random effects` and for the `Residual error parameters`. It is usually recommended to run SAEM with different initializations and to compare the results (see *convergence assessment tool* in Section 3.7.7).

A right click on an initial value displays a list with three options. The default choice is `Estimate` for maximum likelihood estimation. The choice of the "`Fixed`" option (initial values in pink), means that this parameter must be kept to its initial value and so, it will not be estimated. Use the `Priors` option (initial values in blue) to specify a prior distribution for this parameter.
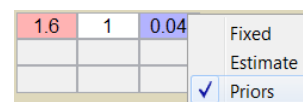
Only the two options `Estimate` and `Fixed` are available for the variances and residual error model parameters

You can also switch between to estimate standard deviations or variances
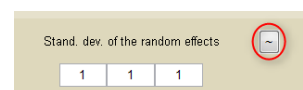
3.4.1 Check initial fixed effects
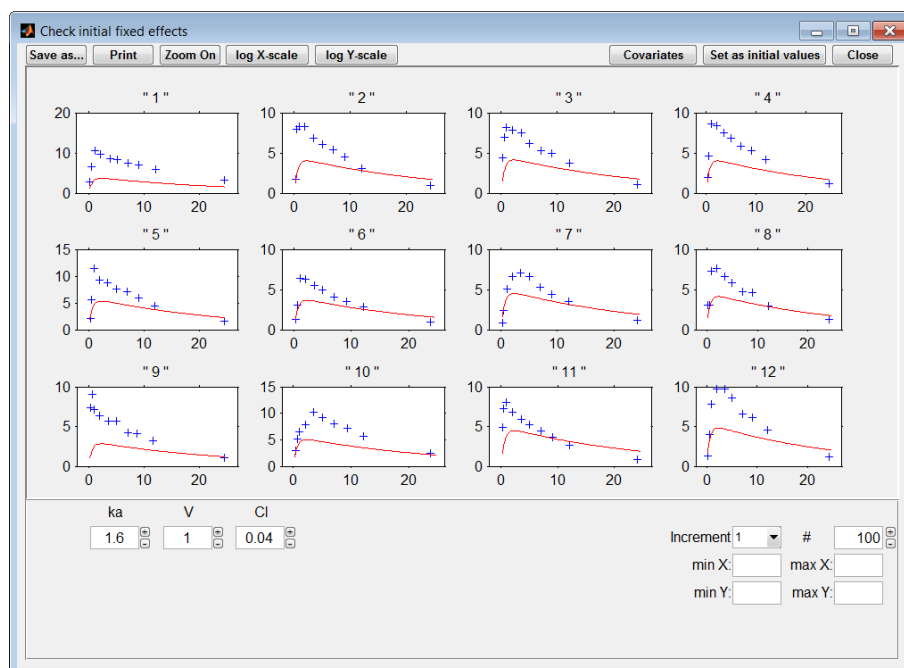
The fits obtained with the initial fixed effects are displayed for continuous observations. It can be useful in case of complex models (e.g. defined with differential equations), in order to find some "good" initial values. You can change the values of the parameters with the buttons + and - and see how the fits change.

3.4.2 Use the last estimates

If you have already estimated the population parameters for this project, then you can use the $\boxed{\texttt{Use the last estimates}}$ button to use the previous estimates as initial values.

3.5 The "Algorithm" frame

There are several settings that control the behavior of the algorithms and the results graphics and tables generation (see Section 3.13). The main interface gives access directly to some of them:

- `Seed:` The default value of the seed used for the random numbers generator is 123456. This seed can be randomly generated with the $\boxed{\texttt{New seed}}$ button.

- **Number of iterations:** There are two stages in the population parameters estimation algorithm. Use K_1 and K_2 to define the number of iterations for each one.

  If `auto` is selected, then algorithm will determine automatically if it can jump from one stage to the other or if it can finish the estimation process. In this case, the numbers K_1 and K_2 will represent the maximum number of iterations for each stage. Here, these numbers will not exceed 500 and 200.

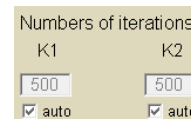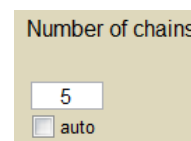- **Number of chains:** When the dataset is small, just a few of subjects for instance, replicating the dataset can improve the precision of the estimation. The number of (Markov) chains specifies how many times the dataset need to be replicated.

  You can specify this number yourself,
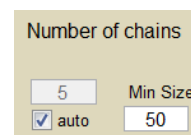
  or you can let MONOLIX to compute how many replicates are needed to ensure a minimum number of subjects. In this case, you should specify this minimum size and any time you change the dataset, the number of chains will be updated according to the number of subjects present in the new dataset.
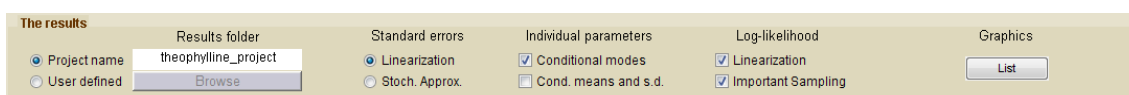
  For instance, in the *theophylline example* there are only 12 subjects in the dataset. Setting to 50 the minimum number of subjects, MONOLIX choose to use 5 Markov chains.

- **Simulated annealing:** A Simulated Annealing version of SAEM is used to estimate the population parameters (the variances are constrained to decrease or increase slowly during the first iterations of SAEM, see Section 3.13.1)

- **Monte-Carlo Sizes:** It can be specified the number of simulated samples used to compute *Prediction distribution* graphic, the *NPDEs* (Normalized Prediction Distribution Errors) and the *VPCs* (Visual Predictive Checks), the log-likelihood estimation when the *Importance Sampling* algorithm is used (see Section 3.7.4)

- **Display:** The number of iterations between two updates of the convergence graphics produced by the algorithm (e.g. SAEM convergence window when estimating population parameters).

3.6 The "Results" frame

- The `Results folder` is the folder where all the results are stored.

  - `Project name`: the result folder is determined automatically from the name of the project as "`<project folder>/<project name>/`".
  - `User defined name`: the name and the directory of the results folder are defined by the user.

- `Standard errors`: Which algorithm to use when computing Fisher Information Matrix and standard errors. Can be `by Linearization` (using a linear approximation of the model) or `by Stochastic Approximation` (the observed Fisher Information Matrix of the exact model is estimated by stochastic approximation).

- `Individual parameters`: says if the `Conditional modes` and/or `Conditional means and standard deviations` should be computed when estimating the individual parameters

- `Log-likelihood`: says which algorithms to use when estimating the log-likelihood. Can be `by Linearization` (using a linear approximation of the model) and/or `by Importance Sampling` (the log-likelihood of the exact model is estimated by Monte-Carlo).

- `Graphics`: Use $\boxed{\text{List}}$ button to choose the list of graphics to produce when the results are computed.

3.7 Executing tasks

MONOLIX includes several estimation algorithms: estimation of the population parameters, the Fisher information matrix and standard errors, the individual parameters, and log-likelihood. Also, different types of results are available in the form of graphics and tables.

For illustration, we will consider a project with the theophylline data as represented in the Figure above. Here, the log-weight centered by the mean is used as a continuous covariate and the gender as categorical.

3.7.1 Estimation of the population parameters

Maximum likelihood estimation of the population parameters is performed with the button.

The sequence of estimated parameters (θ_k) is displayed in a figure which is automatically saved as `saem.png` in the results folder at the end of the estimation. Also, the final estimations are displayed in the MATLAB command window.

This algorithm produces the file `pop_parameters.txt` in the result folder, with all the estimated parameters: population parameters, variances (or standard deviations), correlations, error model parameters, etc. It also includes the project filename, the date and hour of the run, and the version of MONOLIX . The estimation of the population parameters by groups defined by the categorical covariates used in the model are also given.

The same information is printed in the screen (MATLAB command window for MATLAB version and DOS or linux terminal for standalone version). The algorithm convergence graph (`saem.png`), as well as a copy of the used project in `.mat` and `.xmlx` formats are also saved. Also, if the structural model is coded using MLXTRAN, a copy of this file is included.

▷ *theophylline example:*

The final estimations for our example, and the convergence graphics for this exemple are:

We clearly see on these graphics that the trajectory of (θ_k) is much more random during the first stage than during the second stage. The number of iterations used for the SAEM algorithm were $K_1 = 173$ and $K_2 = 120$.

Remark 1: This figure shows that, despite a very poor initial guess, SAEM algorithm converges in very few iterations to the neighborhood of the solution. Thus, in this example, a good estimation of the parameters can be obtained with very few iterations of SAEM (try for instance with $K_1 = K_2 = 20$ and any initial guess...). Also simulated annealing is not necessary for this project.

```
Estimation of the population parameters

                     parameter
ka                 :      1.58
V                  :      0.455
beta_V(t_WEIGHT)   :     -0.423
Cl                 :      0.0438
beta_Cl(SEX_M)     :     -0.157

omega_ka           :      0.651
omega_V            :      0.12
omega_Cl           :      0.253

a                  :      0.731


_____
Estimation of the population parameters by groups

                  parameter
Cl_(SEX=F*)  :     0.0438
Cl_(SEX=M )  :     0.0374


Numerical covariates
    t_WEIGHT = log(WEIGHT)       Centered by the mean ( log(mean(WEIGHT)) = 4.2425 )
Categorical covariates
    SEX
        Reference group: F
        Other groups:  M
```

However it is not always the case. One can see in the figure below that the choice of the first number K_1 of iterations is crucial. In this example, the total number of iterations is $K_1 + K_2 = 100$, with $K_1 = 50$ on the left and $K_1 = 1$ on the right. The second example shows that $K_1 = 1$ iterations is not enough to reach the neighborhood of the solution. Then, because of the averaging during the K_2 next iterations, SAEM will require many more iterations to converge than in the first example.

Remark 2: The individual parameters of each subject are "roughly" estimated during the last iterations of SAEM (by estimating the conditional means $E(\phi_i|y; \hat{\theta})$). These "rough" estimates are used for the results if the individual parameters are not better estimated later.

3.7.2 Estimation of the standard errors

Computes Fisher information matrix and so the standard errors of the different estimators and their correlation. There are two different algorithms: `by linearization` where the structural model is linearized, and so the full statistical model is approximated by a gaussian model, or `by Stochastic approximation` where the exact model is used, and the Fisher information matrix (F.I.M) is approximated stochastically (slower but more precise).
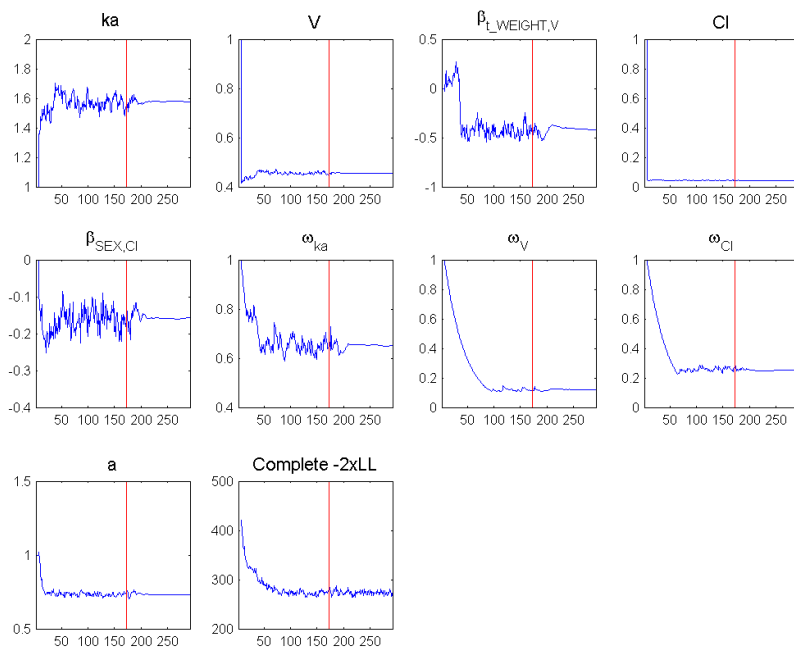
A "nice convergence" of SAEM	A "poor convergence" of SAEM
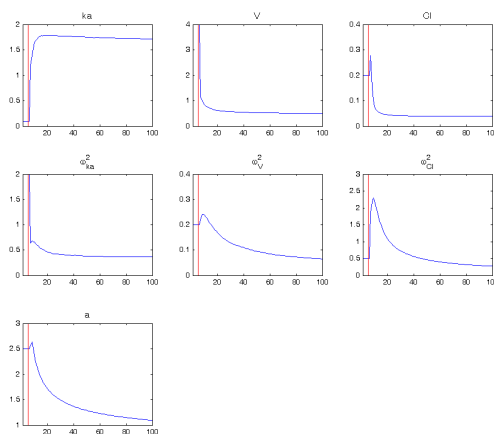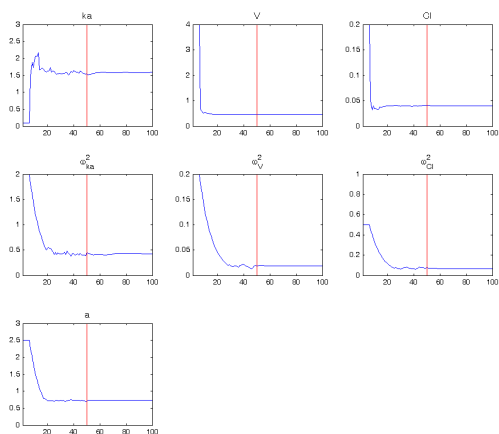
The final estimations are displayed in the command MATLAB window together with the population parameters:

1. the estimated fixed effects, their standard-errors, the absolute and relative p-values obtained from the Wald test (only for the coefficients of the covariates),

2. the estimated variances (or standard deviations) and their standard-errors,

3. the estimated residual error parameters and their standard-errors,

4. the estimated correlation matrix of the random effects if the covariance matrix is not diagonal,

5. the correlation matrix of the fixed effect estimates, with the smallest and largest eigenvalues,

6. the correlation matrix of the variance components estimates, with the smallest and largest eigenvalues,

▷ *theophylline example:*

```
Estimation of the population parameters

                     parameter     s.e. (lin)   r.s.e.(%)   p-value
ka                :      1.58          0.31          20
V                 :      0.455         0.02           4
beta_V(t_WEIGHT)  :     -0.423         0.31          74        0.18
Cl                :      0.0438        0.0055        12
beta_Cl(SEX_M)    :     -0.157         0.16         104        0.34

omega_ka          :      0.651         0.15          22
omega_V           :      0.12          0.036         30
omega_Cl          :      0.253         0.063         25

a                 :      0.731         0.056          8


_____

correlation matrix of the estimates(linearization)

ka                    1
V                   0.14      1
beta_V(t_WEIGHT)      0     0.04       1
Cl                 -0.04    -0.1     0.09      1
beta_Cl(SEX_M)       -0       -0    -0.12    -0.76      1

Eigenvalues (min, max, max/min): 0.24  1.8  7.6

omega_ka      1
omega_V    -0.02      1
omega_Cl     -0    -0.02       1
a          -0.03    -0.1    -0.06      1

Eigenvalues (min, max, max/min): 0.87  1.1  1.3


_____

Estimation of the population parameters by groups

                parameter     s.e. (lin)   r.s.e.(%)
Cl_(SEX=F*) :    0.0438         0.0055         12
Cl_(SEX=M ) :    0.0374         0.004          11
```

All that information is appended to the file `pop_parameters.txt`.

Remark: The Wald test indicates here that $\beta_{W,V}$ and $\beta_{S,Cl}$ are not significantly different from 0: we can consider that the log-volume is not a linear function of the log-weight, and that the clearance of the males and females are not significatively different (of course any statistical inference with only 12 subjects can be dubious...).

3.7.3   Estimation of the individual parameters

Although the population parameter estimation algorithm gives a rough estimation of the individual parameters, it can be estimated two more precise estimators: the conditional mode and the conditional means. Those estimators can be computed with the ⋀ button.

If the option `Estimate the conditional modes` is selected, the individual parameters are estimated by maximizing the conditional probabilities $p(\phi_i|y_i; \hat\theta)$.

If the option `Estimate the cond.  means and s.d.` is selected, the conditional means and standard deviations are estimated by MCMC. For each parameter, the mean of these quantities over all the subjects is displayed together with a r_{mcmc} interval.

Two files `indiv_parameters.txt` and `indiv_eta.txt` are created with the estimated individual parameters and random effects in table format. Also, if there were defined priors on some fixed effects, and it was selected prior distribution method for some of them, a new file called `simulatedPopParams.txt` is created with simulations using their posterior distribution laws.

▷ *theophylline example:*

Here, $r_{mcmc} = 5\%$ and
$L_{mcmc} = 50$
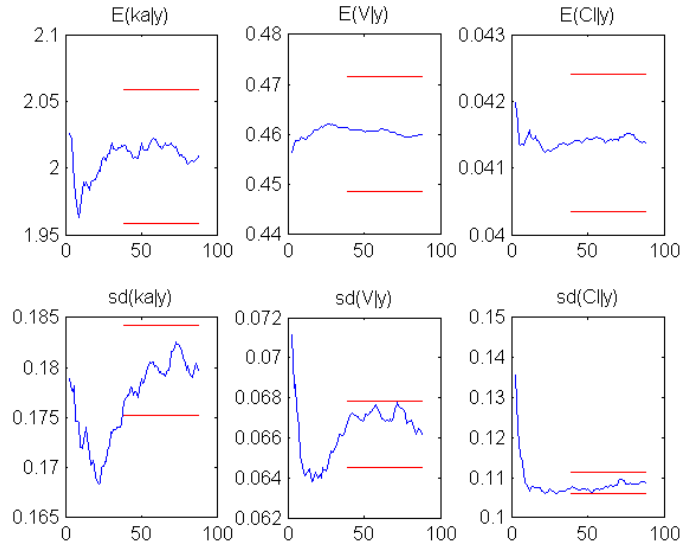
3.7.4 Estimation of the log-likelihood

The estimation of the log-likelihood is performed with the ▥ button. Two different algorithms are proposed to estimate the log-likelihood: by `linearization` and by `Importance sampling`.

▷ *theophylline example:*

In our example, the two different estimates of the log-likelihood are computed.

The value of the two estimated $-2\times$ log-likelihoods, the standard error of the Monte-Carlo estimate, The AIC and the BIC are displayed in the command MATLAB window:

```
Log-likelihood Estimation by linearization

-2 x log-likelihood:                    340.41
Akaike Information Criteria    (AIC):   358.41
Bayesian Information Criteria (BIC):    362.78


Log-likelihood Estimation by important sampling
Sampling distribution for the random effects: t with 5 d.f

-2 x log-likelihood:                    341.61 (0.064)
Akaike Information Criteria    (AIC):   359.61 (0.064)
Bayesian Information Criteria (BIC):    363.97 (0.064)

Elapsed time is 3.51 seconds.
CPU time is 9.73 seconds.
```

The estimated log-likelihoods are appended to `pop_parameters.txt`.

Remark: The log-likelihood can not be computed by `linearization` for discrete outputs (categorical, count, etc.) nor for mixture models or when the posterior distribution have been estimated for some parameters with priors.

The Monte-Carlo estimator converges to the observed log-likelihood when the size of the Monte-Carlo increases. The sequence of estimates, as a function of the Monte-Carlo size, is displayed as a Figure:
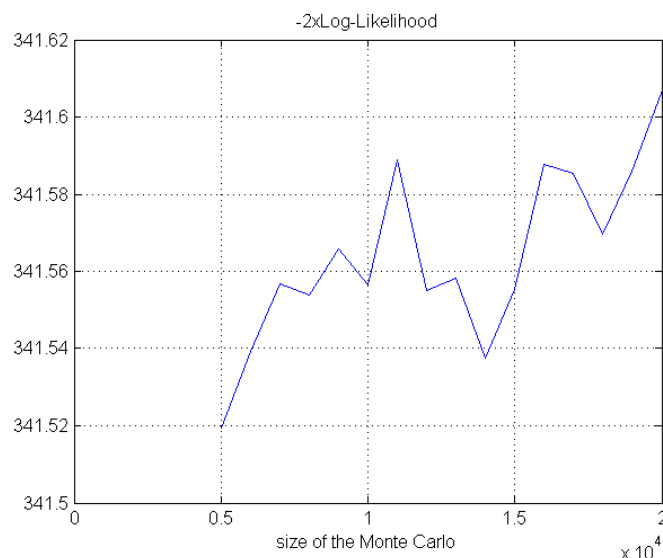
3.7.5 Computing results

Use the button ![icon] to compute produce some graphics and tables.

It is possible to specify the list of graphics (with button ⟨List⟩ at the bottom of the main interface), which of them should be saved and which tables should be produced (menu `Graphics->Outputs to save`). For more details see Section 3.8.

3.7.6 Running several algorithms

With the button ⟨Run⟩, you can executes successively several tasks. The list of tasks to run is called "scenario" or "workflow". You can define your own scenario by checking the corresponding checkboxes at the left of the ⟨Run⟩ button.

For example, define the following workflow
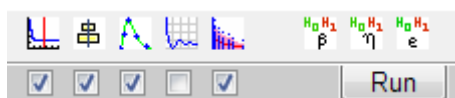
in order to:

1. estimate the population parameters,

2. estimate the standard errors,

3. estimate the individual parameters,

4. display graphical results using the individual parameters estimated previously.

If you just want to compute the population parameters with their standard errors, define the workflow:
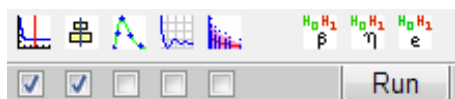
3.7.7 Algorithms convergence assessment

MONOLIX includes a convergence assessment tool. It is possible to execute a workflow as defined above but for different, randomly generated, initial values of fixed effects.

Click in the $\boxed{\texttt{Assessment}}$ button on the workflow bar to open the Assessment graphical interface.
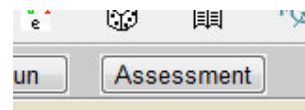
You can enter the number of runs, or replicates, the parameters to generate initial values and the intervals where those values should be (uniformly) simulated. Click on $\boxed{\texttt{Run}}$ to execute the tool.
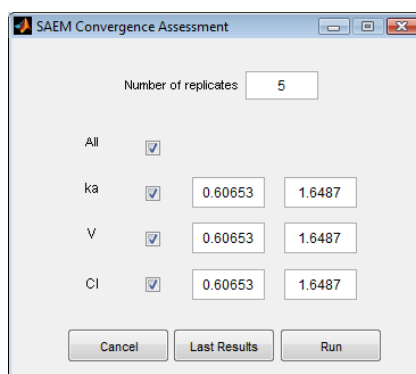
Remarks:

- The project workflow is used (see Section 3.7.6).

- Population parameters estimation must be selected.

- A result folder is generated for each set of initial parameters.

Two kinds of graphics are given as a summary of the results,

  one showing the estimated values (blue) with the estimated standard errors (red bars) for each replicate (if the Fisher information matrix estimation was included in the scenario)
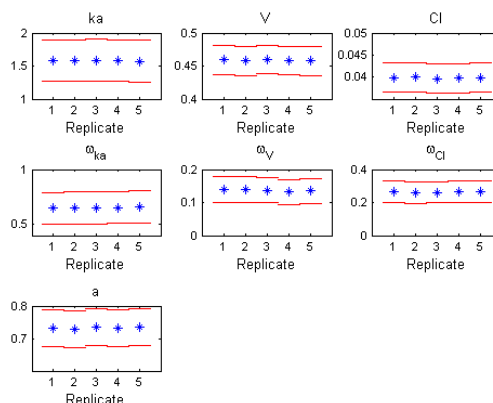
  and the superimposed convergence graphics for each parameter. In case of F.I.M estimation is included, then the black lines represents the mean of the estimated s.e.
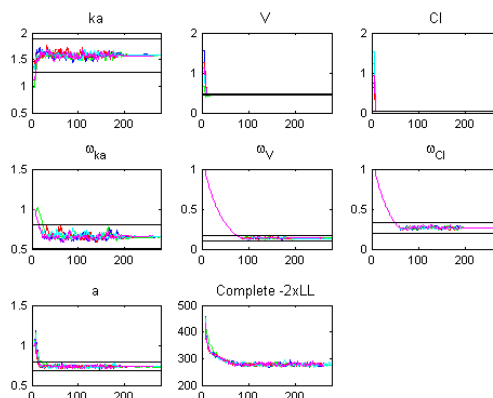
  Also, a `.mat` file is produced with all the results used for those graphics. Press `Last Results` button to reproduce the last result graphics if you already have ran the tool for the current project, and `Cancel` to close the interface.

3.8 Plots and results

  From the main interface toolbar, you can

- display the observed data y as a function of the regression variable (e.g. `time` for a PK applications):

- Produce several graphics and tables from the results of the algorithms:

  You can select which figures to show in menu `Graphics->List` or button $\boxed{\text{List}}$ in the interface (see Section 3.8.4) and which figures and/or tables to save (Section 3.8.6)

  You can also produce one specific graphic.

Note: By default the new projects do not save the graphics and do not produce the tables. If you want MONOLIX to create some table, you must select it in menu `Graphics->Output to save`.

MONOLIX allows to change some visual preferences determining the way the graphics are shown (set of line colors and styles, markers, etc). It is also possible to specify the format where the graphics should be saved. Section 4.14 shows how you can set some preferences with an interface and Appendix B gives an overview of all the parameters that can be set and how.

3.8.1 The graphics

The proposed figures are separated in three groups:

- `Reduced`: Graphics rapid to create, good to have a first view of the goodness of the fits.

- `Simulation`: Graphics requiring simulations, so the can take some time to be produced. Includes some Visual Predictive Checks (VPC) graphics.

- `Others`: other graphics proposed.

Reduced :

`Project Summary`: displays some information about the project (date, datafile,...)

15-Oct-2012 09:52:15

Output   : concentration
Data file   : theophylline_data.txt
Structural model: oral1_1cpt_kaVCl
Results folder   : theophylline_project

Project path : C:
Data path   : C: monolixData\monolix420\work\Demos\theophylline
Model path : D: matlab\libraries\PKLibrary
Results path: C: Desktop

Spaghetti plot: displays the original data as a "spaghetti plot".
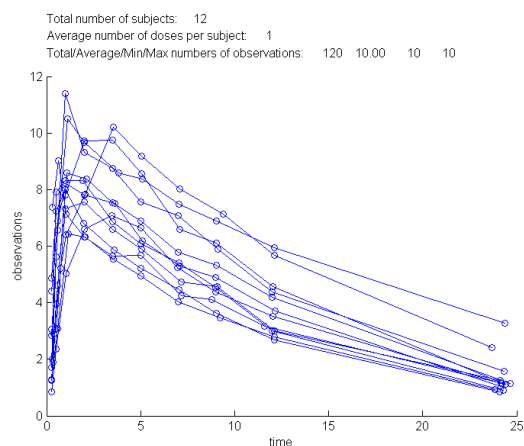
Individual fits: displays the individual fits, using the population parameters with the individual covariates (red) and the individual parameters (green) on a continuous grid.

It is also possible to display the median and a confidence interval for (y_{ij}) estimated with a Monte Carlo procedure. The design and the covariates of each subject are used for the simulation.
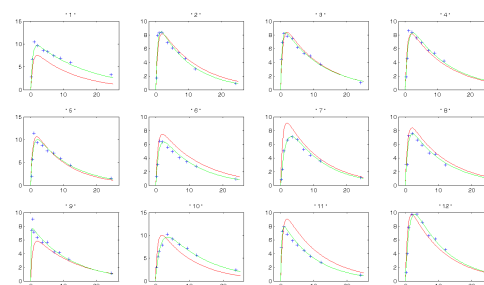
Obs. vs Pred.: displays observations versus the predictions $(\hat{y}_{ij})$ computed using the population parameters (on the left), and with the individual parameters (on the right)
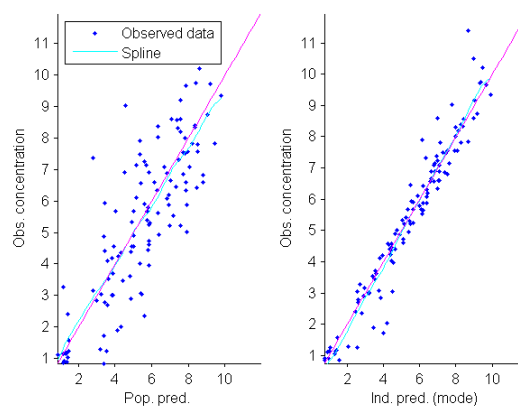
Covariates: displays the estimators of the individual parameters in the gaussian space, and those for random effects, (e.g. the conditional expectations $\mathbb{E}(\varphi_{i\ell}|y;\hat{\theta})$ and $\mathbb{E}(\eta_{i\ell}|y;\hat{\theta}), 1 \leq i \leq N$ and the conditional modes) *v.s.* the covariates.
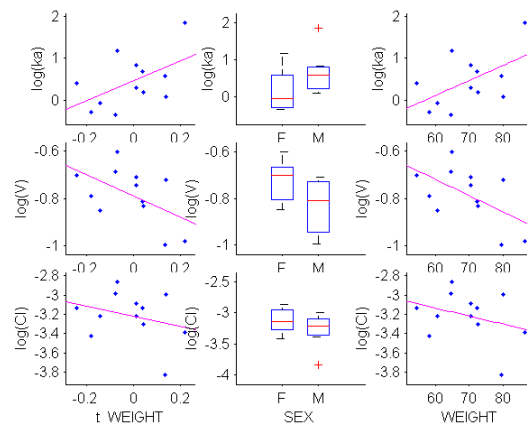
Parameter Distributions: displays the estimated population distributions of the individual parameters. Settings allows you to show a summary of the theoretical distributions (mean, median, mode, quartiles, percentiles, standard-deviation), histograms, and a non-parametric estimation of the distribution
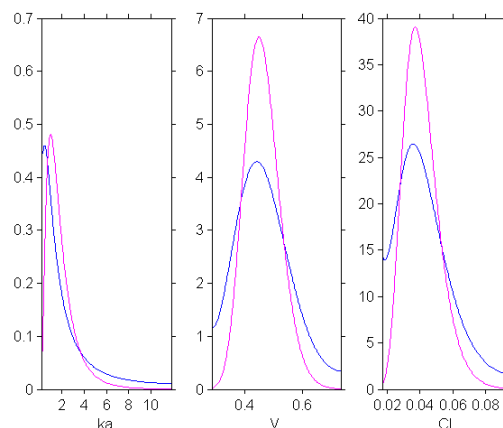
It is also possible to display the empirical distribution of the individual parameters and the shrinkages computed as follows for each random effect:

$$\text{Shrinkage} = 1 - \frac{\text{Var}(\hat{\eta})}{\hat{\omega}^2}$$

Monolix 4.2.2
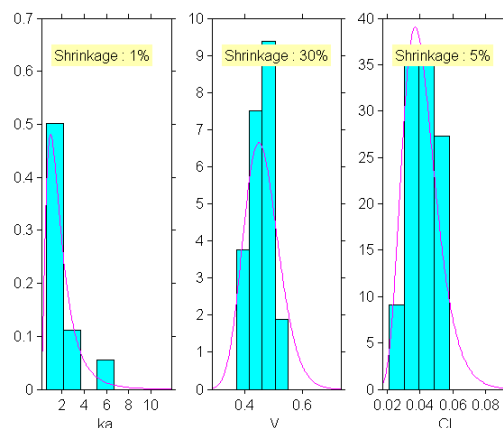
Random Effects(boxplot): displays the distribution of the random effects with boxplots. The horizontal dotted lines show the interquartile interval of a standard Gaussian distribution.
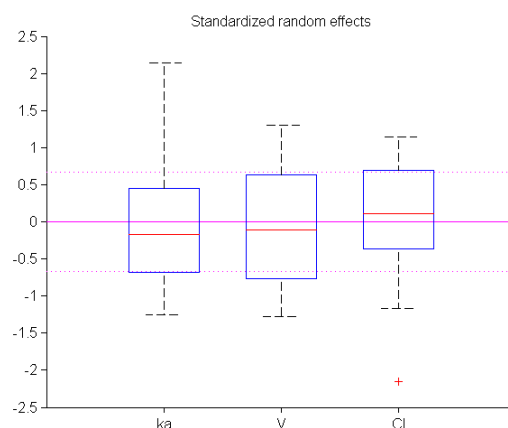
Random Effects(joint dist.): generates scatter plots for each pairs of random effects.
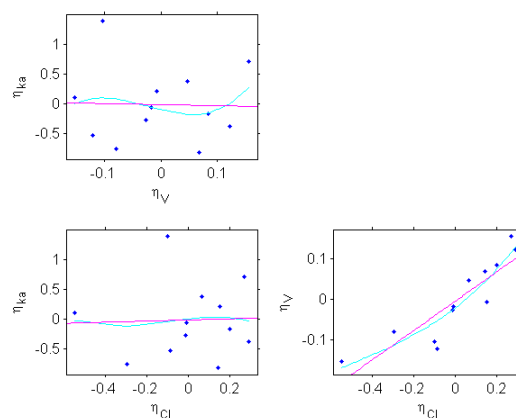
Convergence SAEM: displays the sequence of the estimated parameters (θ_k).
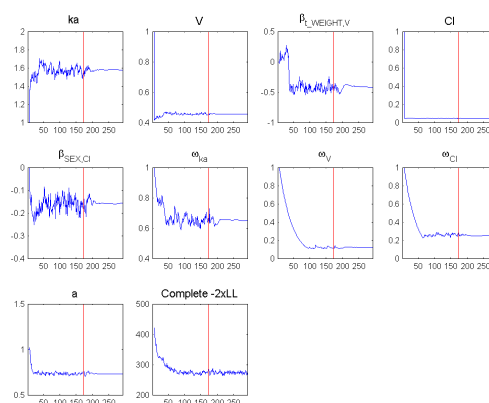
Simulations :

Residuals: displays the PWRES (population weighted residuals), the IWRES (individual weighted residuals) and the NPDE (Normalized Prediction Distribution Errors) v.s. x (top) and v.s. the predictions (bottom).

The PWRES are computed using the population parameters and the IWRES are computed using the individual parameters.
For discrete outputs, only NPDE and individual NPDE are used.
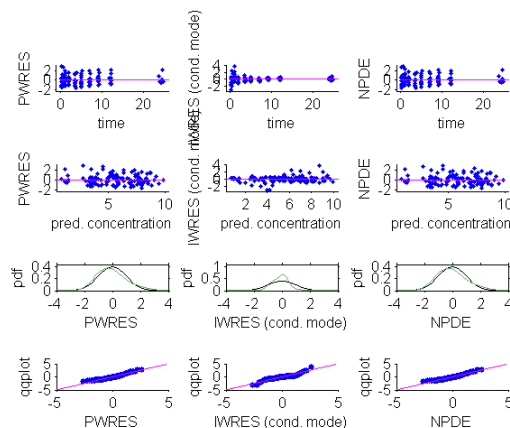
For continuous outputs, this figure shows the empirical distributions of the weighted residuals and the NPDE together with the standard Gaussian pdf and qqplots to check if the residuals are Gaussian.

Diagnostic VPC: Allows to compare the empirical distribution of the observations and theoretical distribution (computed from simulations in observation times) inside some parameterizable bins.
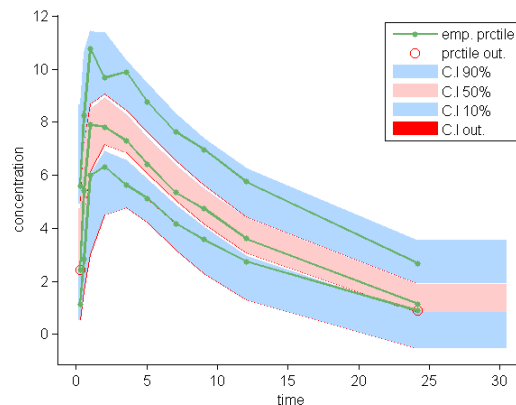
NPC: Allows to compare empirical cumulative distribution function (CDF) of the observations with theoretical's (computed from simulations).
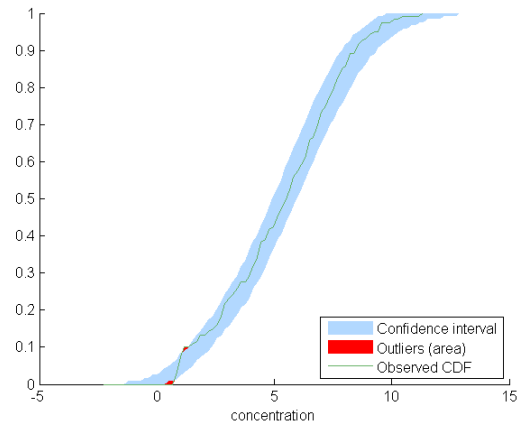
BLQ: Shows the proportion of censored data on time. It is possible to chose the censoring interval. This graphic is only available for projects with censored data.
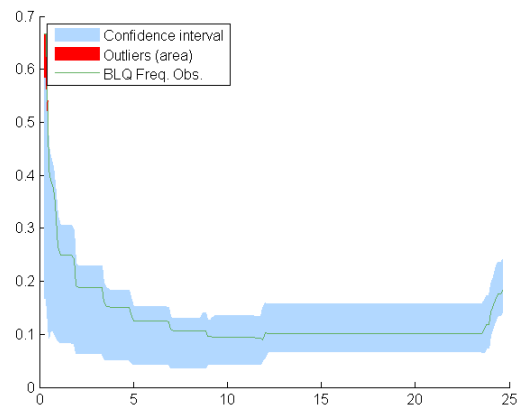
Prediction distribution: Allows to compare the observations with the theoretical distribution of the predictions.
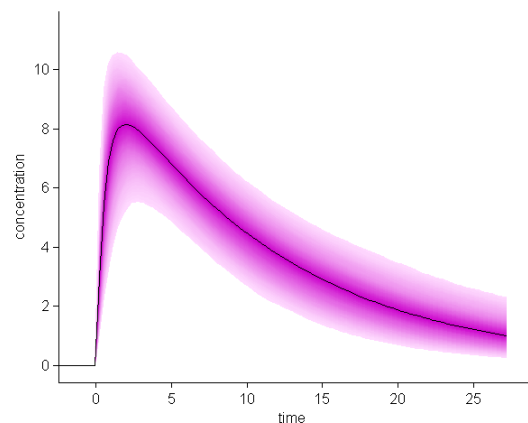
Others :

Categorized data: Allows to check the distribution of the observations in categories that can be parameterized by the user.
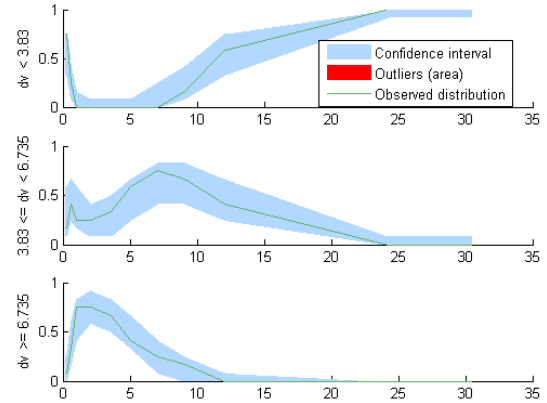
Time to Event data: Shows empiric (from observations) and theoretical (from simulations) survival function and average number of events for *Time to event* data. Only available for outputs of type event.

LIXOFT
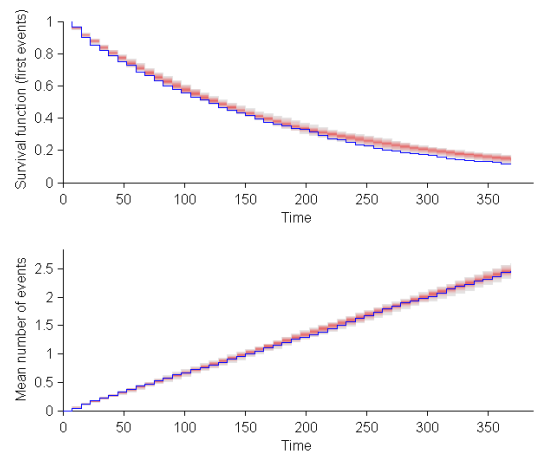
`Transition Probabilities`: For discrete observations, shows the proportion of each category knowing the previous one. Can be used on any discrete model.
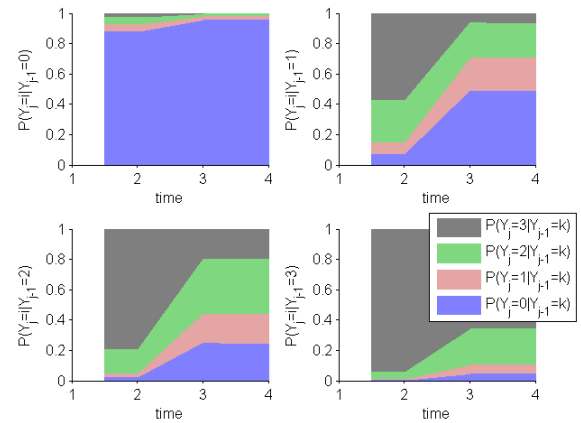
`Posterior Distribution`: Shows the priors (theoretical) and the posterior (simulations) distributions for each parameter estimated by "posterior distribution". Requires the estimation of the `conditional means` of the individual parameters.
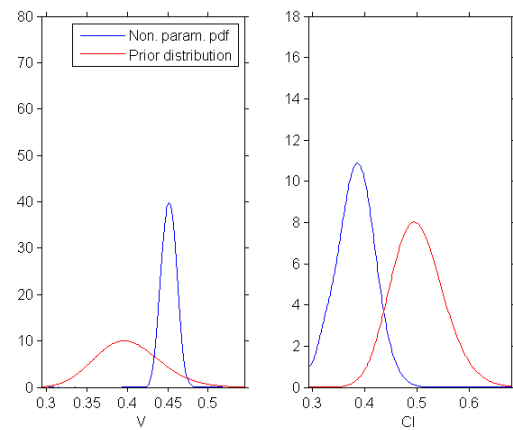
`Contribution to log-likelihood`: Displays the contribution of each subject observations to the total log-likelihood. Requires the estimation of the log-likelihood, and both estimators are shown if they have been computed.
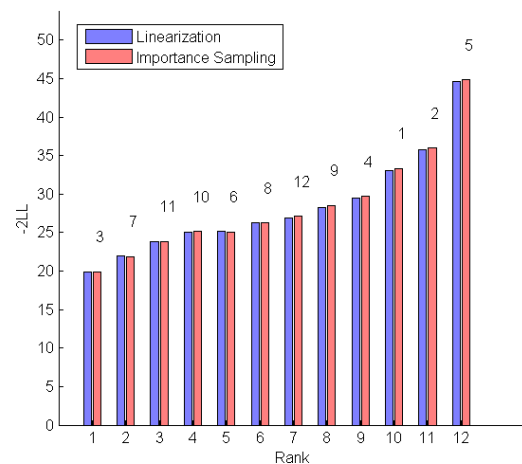
3.8.2 The tables
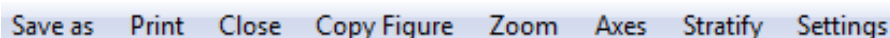
MONOLIX can produce five kinds of `ASCII` table files:

- `Observation times` contains the regression variables, the individual predictions, the population predictions, the weighted population residuals, the weighted individual residuals and the NPDE, for all the observation times in the dataset.

- `Fine-grid times` contains the regression variables, the individual predictions, the population predictions in a fine (regular) grid.

- `All times` contains the regression variables, the individual predictions, the population predictions for a possibly larger set of times in the dataset, including observation times, lines where `MDV` column is set to 2.

- `LL individuals contribution` contains subject contribution to the total log-likelihood.

- `Covariates summary` contains a summary of all the covariates defined in the project.

The first three tables can include new columns according to the `table` definition in the structural model as described in Section 4.11.

3.8.3 The graphics menu bar

The graphics menu bar proposes different options

- It is possible to save, close or print the figure and, in Windows, to copy and paste it to any other document.

- `Zoom` menu activates or de-activates the zoom

- The `Axes` menu proposes to use semi-log or log-log scale plots. `Settings` option gives the opportunity to choose labels and to change the scale of graphics. It is also possible to use the same axis limits for the different plots of the same graphic.

- The `Stratify` menu opens the stratify window, see Section 3.8.5 below.

- The `Settings` menu opens a new window with the settings of the current graphic.

3.8.4 Main interface Graphics Menu

This menu options allows to handle the graphics configurations of the project, which are used each time the results figures and tables are launched. Those configurations have three parts: the list of result graphics to produce, each graphic settings and the list of graphics to save and tables to produce.

There are 4 options in Monolix `Graphics` menu.

- `Attach the current settings to project`

- `List`

- `Settings`

- `Outputs to save`
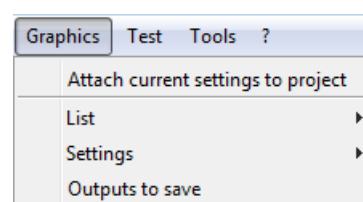
Attach the current settings to project

When this option is chosen, the current settings of the opened figures are used as current project defaults.

List

This menu allows to manage the list of graphics to display.

Three options are available :

- `Load`
  Allows to load already defined lists. Those lists are divided in two section: Monolix predefined lists and user defined lists.
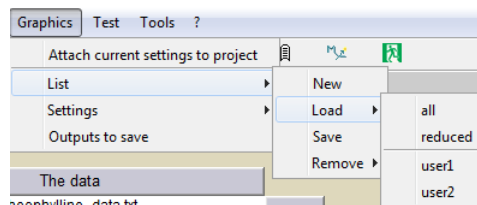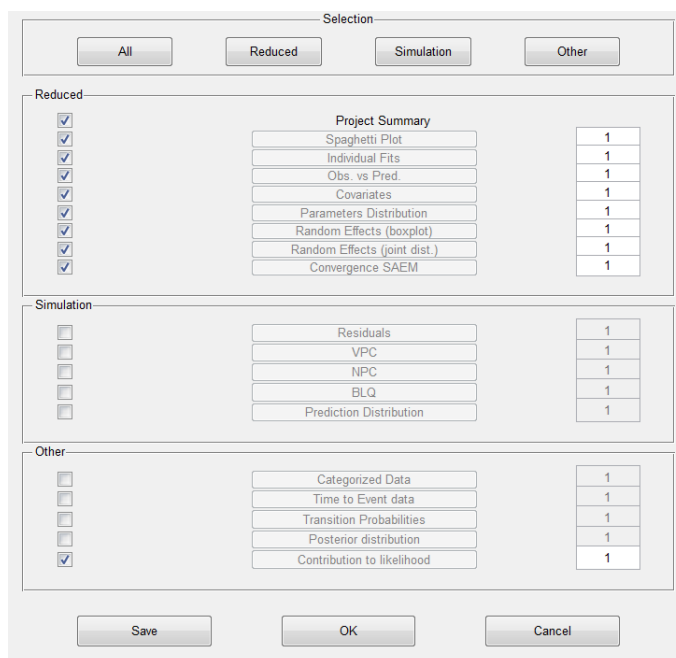
The chosen list determines the figures that will be opened next time the results graphics are launched (button).

- `New`
  Allows to define and save a list.
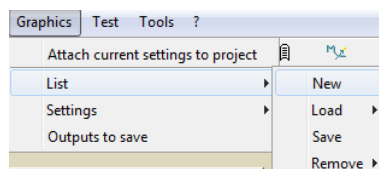
This option opens a window that allows to create a list and then either to export to a `.xmlx` file or to use in the current project.

MONOLIX proposes four predefined lists: `Reduced`, `Simulation`, `Others` as explained in Section 3.8.1, and `All`. The default list is `reduced`.

To define the list, it should be selected which graphics to show for each output (checkboxes at the left, each column represents an output) and how many times (at the right).

It is possible to launch a single graphic by clicking on the corresponding button.

To export the defined list, use the *Save* button. The list should be saved in the proposed folder to be used by MONOLIX.

In order to use the defined list in the current project, click on the button *OK*.

Remark : This window is available in MONOLIX main interface.

- Save
  Saves project's graphic list

- **Remove**
  Remove an user defined list.

Settings

Allows to handle predefined graphic configurations : load, save and remove.

- **Load**

  Allows to load already defined configurations. Those configurations are divided in two section: MONOLIX predefined configurations and user defined configurations.

  The opened graphics are automatically updated when a configuration is loaded.
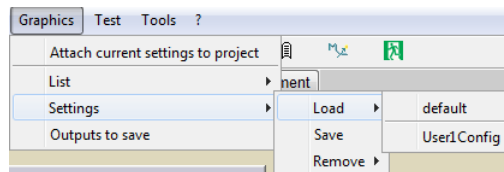
- **Save**
  Allows to save a current configuration. The saved file will be included to the **Load** list.
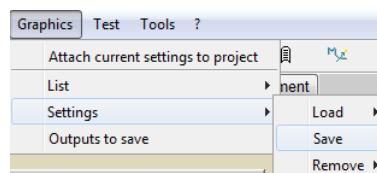
Enter a name to save the configuration.
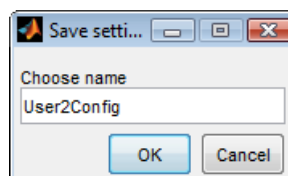
- **Remove**
  Remove an user defined configuration.
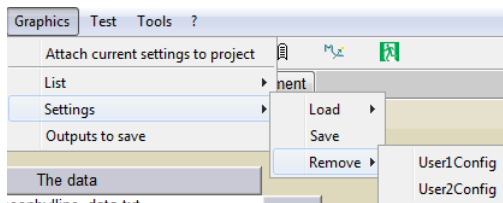
3.8.5   Stratify

The `Stratify` menu allows to create and use covariates for stratification purposes. It is possible to select a categorical covariate for splitting, filtering or coloring the dataset. Also, in the bottom section, it can be defined a time filter
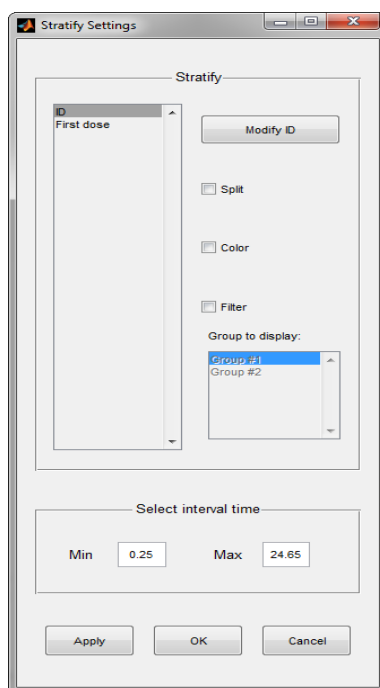
3.8.6   Settings

Each graphic has its own settings, which can be accessed by `Settings` menu. The most common types of options are described below.

Display options

Most of the graphical components can be made visible or hidden, allowing the user to focus on the desired information.

For some graphics, it is possible to select which plots to show.

Prediction Distribution with two different configurations
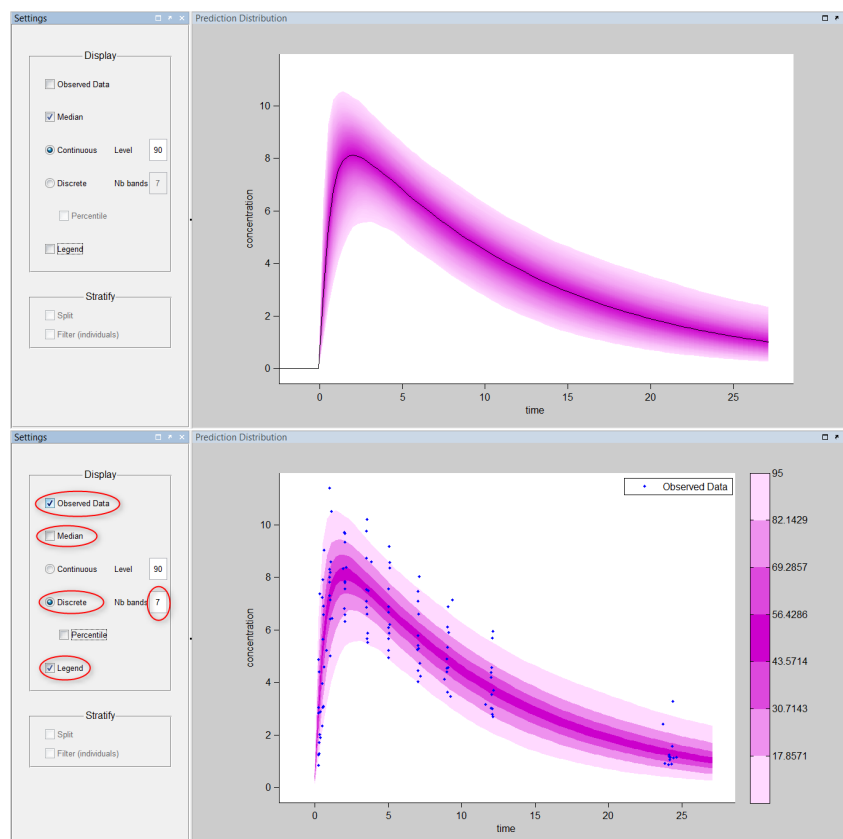
Stratify options

Some graphics can be stratified using the filters issued from the `Stratify` interface mentioned in Section 3.8.5. It is possible to activate the splits, filters, and/or colors for groups of individuals and also to filter observations for some time interval (the first regression variable is used if no time column exists).

Residuals with 2 kinds of residuals with `scatter plots (predictions)` and `QQ-plot` chosen
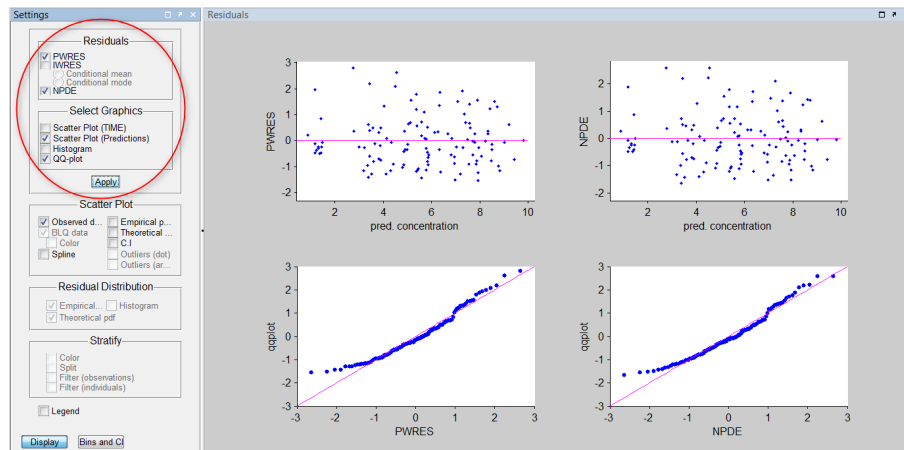
Predictions Vs Observations without applying stratify
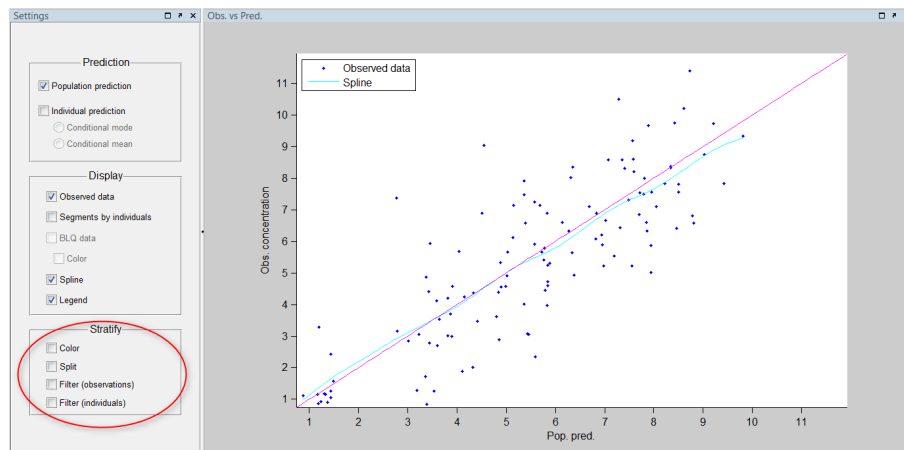
Split: *Predictions Vs Observations* with `Split` option set
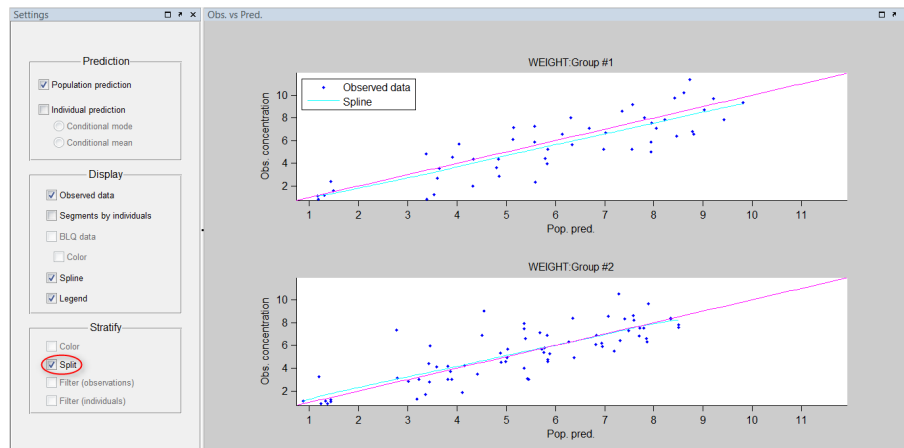
Individuals filter: *Predictions Vs Observations* with `Filter (individuals)` option set
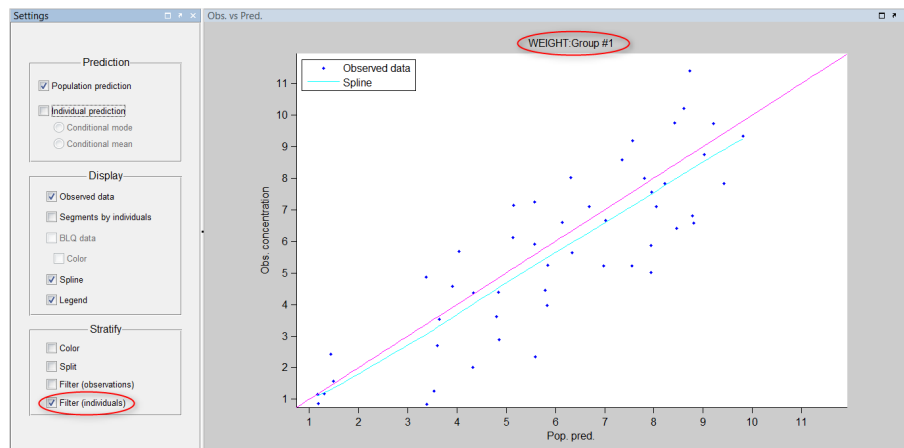
Color: *Predictions Vs Observations* with `Color` option set
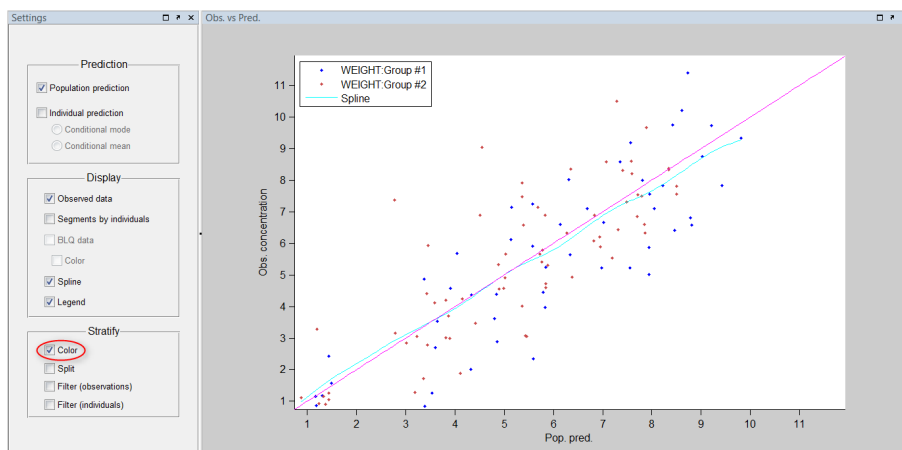
Observations filter: *Predictions Vs Observations* with `Filter (observations)` option set
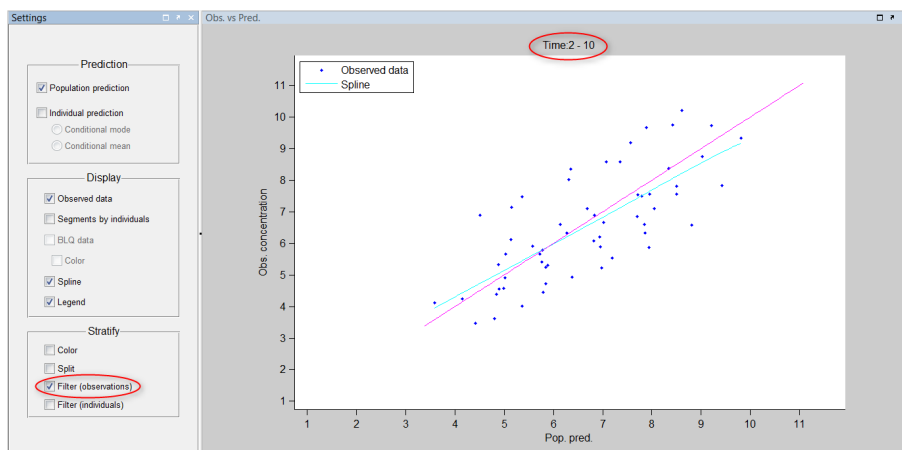
Saving graphics

It is possible to choose which graphics to save. Clicking on `Outputs to save` in `Graphics` menu, a window is opened :
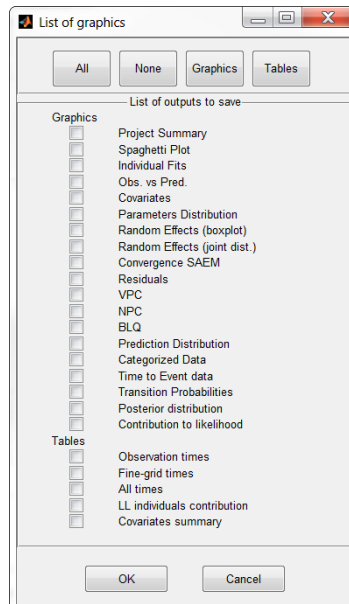
The save starts after the display of the graphics.

Computations options

The user can modify some parameters required to compute the information to be represented. This allows him to tune the graphics.
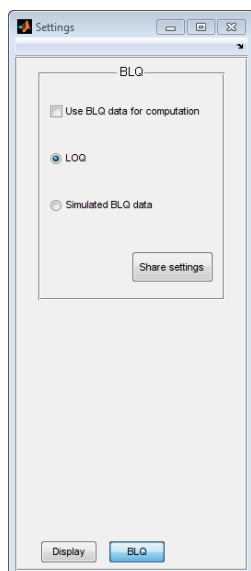
Figure 3.1: BLQ settings
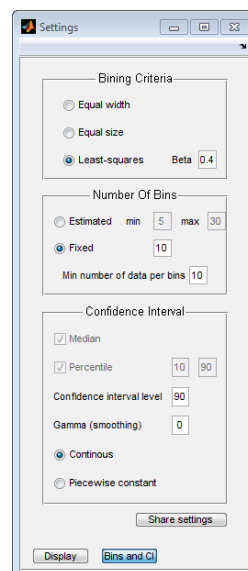
Figure 3.2: Bins settings

Share

Some options are identical for several graphics and can be set independently from one graphic to the other. The button *Share* allows to set the current values to all the graphics sharing those options. The following table describes links between the different graphics. The three columns contain respectively the shared options, the *Share* button name, and the concerned graphics.

Option	Button name	Graphics
BLQ	**Share settings**	Predictions Vs Observations
		Residuals
		VPC
		NPC
BINS	**Share settings**	Residuals
		VPC
		Categorized data
Estimators	**Share selection**	Parameter distribution
		Random effects (boxplot)
		Joint distribution
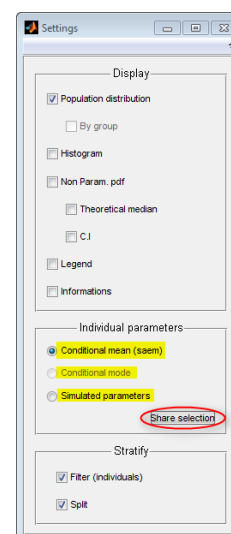
3.9   Testing hypotheses

Three kind of tests can be performed:

- $\substack{H_0 \, H_1 \\ \beta}$ tests the covariate model for the fixed effects,

- $\substack{H_0 \, H_1 \\ \eta}$ tests the covariance model for the random effects,

- $\substack{H_0 \, H_1 \\ \epsilon}$ tests the residual variance model.

The $\substack{H_0 \, H_1 \\ \beta}$ button allows to define the matrix A_0 and A_1 that define the covariate model under hypotheses H_0 and H_1. Consider as an example that we want to test whether the clearance of a subject is function of his weight. Thus, assuming that the absorption rate and the volume are not function of any covariate under both hypotheses, we want to test model H_1 against model H_0 defined as follows:

The other parameters used for the model and the algorithm are those defined in the main Monolix window.

The maximum likelihood estimate and the likelihood are computed under both hypotheses. For both hypotheses, the AIC and BIC criteria are displayed, together with the *deviance* $(-2 \times \log \ell(y; \widehat{\theta}))$. The brackets contain the s.e. of these statistics. The Likelihood Ratio Test (LRT) is performed and the level of significance (*p-value*) is displayed. Here, the log-likelihood is estimated using a Monte-Carlo Importance Sampling algorithm. The Monte-Carlo size is defined as the "LL" size in the Monolix window.

▷ *theophylline example:*
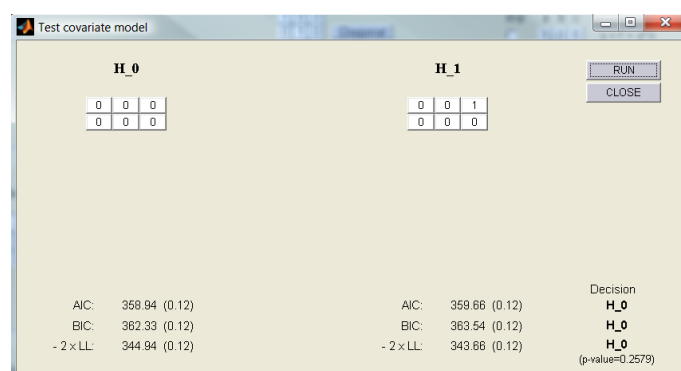
Here, AIC, BIC and the LRT agree with the Wald test to conclude that $\beta_{(WEIGHT,Cl)}$ is not significant (then, there is no significant effect of the weight on the clearance).
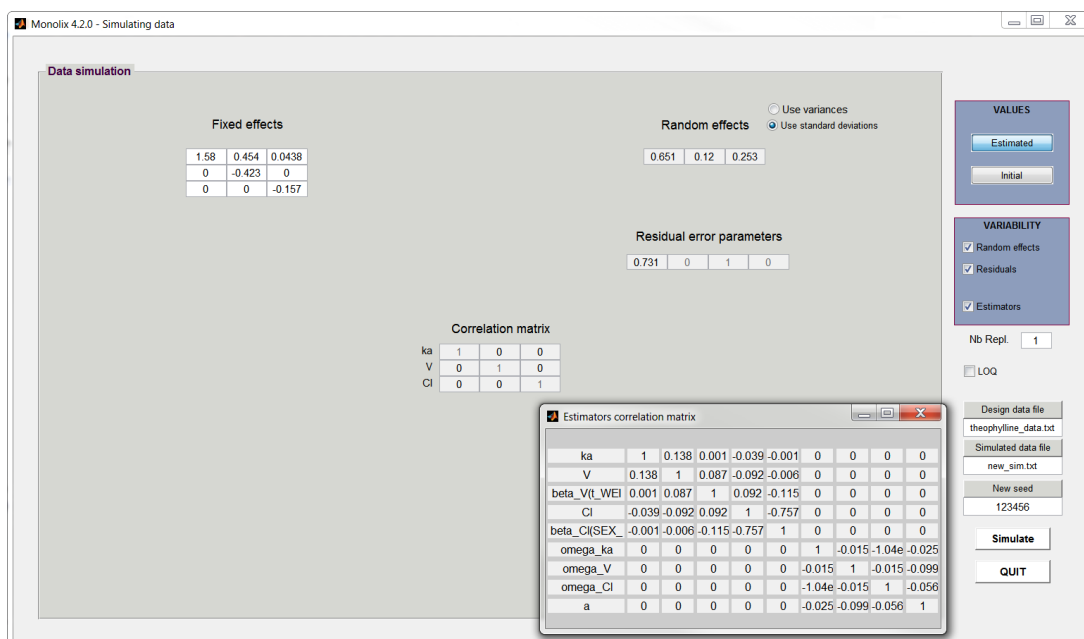
The $\overset{H_0 H_1}{\eta}$ button allows to define the structure of the covariance matrix Ω_0 and Ω_1 of the random effects under hypothesis H_0 and H_1.

The $\overset{H_0 H_1}{\eta}$ button allows to define the residual variance models under hypothesis H_0 and H_1.

A file is generated for each test with the estimations for each hypothesis test and the comparison results.

3.10   Simulation

Use the ⊞ button to open the simulation interface in order to create one or several simulated datasets.

From this interface you can simulate new datasets.

A set of $(\tilde{y}_{ij})$ is simulated using the regression variables (x_{ij}) and the covariates of the original dataset. The simulated data has exactly the same format than in the original data. The only difference is that the observed (y_{ij}) are replaced with the simulated ones $(\tilde{y}_{ij})$.

You are free to set the population parameters yourself or you can use the estimated values, or the initial values given in the main interface.

By clicking on the button $\boxed{\texttt{Design data file}}$, you can also change the design (number of subjects, observation times, covariate values, etc) used for simulations by selecting a new dataset from this interface. However, the chosen dataset must contain exactly the same covariates that were used in the main interface (same name and type among categorical or continuous).

It is also possible to simulate several replicates with the same parameters. All the replicates are stored in the same file and a column `REPL` (for replicate) is added.

You can specify different sources of variability for simulating this dataset: random effects, residual error, uncertainty of the estimates (only if the parameters were estimated).

If you want to simulate with censored data, select `LOQ`. The opened dialog allows to select the censoring intervals. For each output, you can enter one number to specify the `LOQ` so the observations are censored if $Ysim \leq LOQ$, or an interval $[A, B]$ to censor if $A \leq Ysim \leq B$.

3.11   Publishing the outputs

Beware: this feature is not available with the stand-alone version of Monolix.

Use the 📖 button to generate a report on your project, including figures and results.

Use the $\boxed{\texttt{edit}}$ button to modify the Matlab script which selects the different procedures that will be executed.
You can generate your report as a `html`, `doc`, `xml` or `latex` file and all the files will be saved in the results folder.
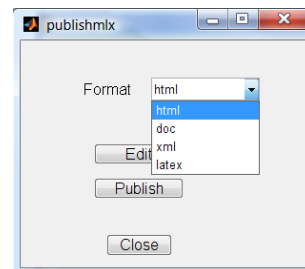
3.12   The results folder

As explained before, several output files are generated by the different tasks. All of them are saved in the result folder chosen by the user (see Section 3.7.5). Here we make a summary of some of those files and when are they generated.

1. With the population parameters estimation (button ⊞):

- `results.mat` is a binary "MAT-file" file that contains all the results

- `project.mat` and `project.xmlx` contain the project respectively in `.mat` and `.xmlx`. Thus, this project can be reloaded with the 📂 button and running the SAEM algorithm with the ⊞ button will reproduce the same results.

- `pop_parameters.txt` contains the estimated population parameters.

- `saem.png` is a PNG file where the sequence of estimates (θ_k) is plotted.

Some of the variables stored in this `results.mat` are:

`fixed_effects`	:	the estimated fixed effects
		(for the log-parameters in the case of log-normal distributions)
`cov_random`	:	the estimated covariance matrix of the random effects
`g_abc`	:	the estimated parameters of the error model

2. With the Fisher information matrix estimation (button ⊞) :

- The results of population parameters estimations are completed with the estimated squared errors (s.e), the relative s.e and the p-value for covariates coefficients as shown in Section 3.7.2 and appended to the file `pop_parameters.txt`.

3. With the individual parameters estimation (button 𝄠) :

- A file `indiv_parameters.txt` which contains the estimated individual parameters is generated: the conditional modes $(m(\phi_i|y;\hat{\theta}), 1 \leq i \leq N)$ and/or the conditional expectation $(\mathbb{E}\left(\phi_i|y;\hat{\theta}\right), 1 \leq i \leq N)$ with the conditional standard deviations $(\sqrt{\text{Var}\left(\phi_i|y;\hat{\theta}\right)}, 1 \leq i \leq N)$ and the covariates.

- A new file `indiv_eta.txt` that contains the estimated random effects is generated: the conditional modes $(m(\eta_i|y;\hat{\theta}), 1 \leq i \leq N)$ and/or the conditional expectation $(\mathbb{E}\left(\eta_i|y;\hat{\theta}\right), 1 \leq i \leq N)$ with the conditional standard deviations $(\sqrt{\text{Var}\left(\eta_i|y;\hat{\theta}\right)}, 1 \leq i \leq N)$ and the covariates.

- The file `simulatedPriors.txt` is created whenever it is estimated the posterior distribution of parameters with priors and the conditional distributions (means and s.d) are estimated. It contains some samples of those parameters simulated following the posterior distribution

4. With the log-likelihood estimation (button 📉):

- the value of the (-2)log-likelihood together with AIC, BIC are added at the bottom of the file `pop_parameters.txt`.

The last three algorithms complete the `results.mat` file with their own results, adding fields like

`se_fixed`	:	the standard-errors of the estimated fixed effects
`pv_fixed`	:	the p-values for the estimated fixed effects
		(for the log-parameters in the case of log-normal distributions)
`se_random`	:	the standard-errors of the diagonal elements of this estimated covariance matrix
`se_g_abc`	:	the standard-errors of the estimated parameters of the error model
`logl`	:	the estimated log-likelihood
`condmode_param`	:	the individual conditional mode of the parameters $\arg\max_\psi \mathbb{P}(\psi|y)$
`condexp_param`	:	the individual conditional expectation of the parameters $\mathbb{E}(\psi|y)$
`condsd_param`	:	the individual conditional standard errors of the parameters

5. With the results (📊) button:

Produced `ASCII` table files if the corresponding tables were selected (see Section 3.8.2):

- `predictions.txt` contains `Observation times` table.

- `finegrid.txt` contains `Fine-grid times` table.

- `fulltimes.txt` contains `Fine-grid times` table.

- `individualContributionToLL.txt` contains `LL individuals contribution` table.

- `covariatesSummary.txt` contains `Covariates summary` table.

Produced graphic files:

Depending on the chosen figure format (see Section 4.14) you will have

- for Postscript (`ps` format):

  - `results.ps` includes a selection of graphics: `spaghetti-plots, residuals, observation vs. predictions`, etc.

  - `individual_fits.ps` includes the individual fits for all subjects, plotted in several pages.

- Any other format:

    - One file for each graphic and several files for `individual fits` graphic, according to the number of subjects in the dataset.
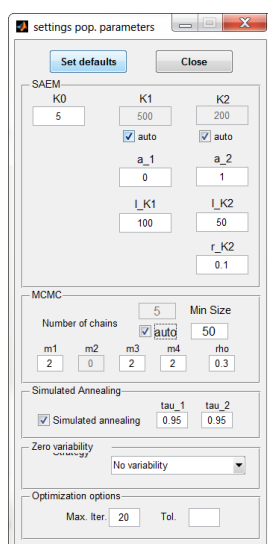
5. Other files:

- The hypothesis tests (buttons $^{H_0\,H_1}_{\quad\beta}$, $^{H_0\,H_1}_{\quad\eta}$, $^{H_0\,H_1}_{\quad\varepsilon}$) generate files with each test results.

- `Publish report` files

- Simulated files

- One `.zip` file for each run containing the corresponding results if `timestamping` has been activated (see Section 4.14).

Note: The graphics are saved like they are produced, i.e using current graphics settings (see Section 3.8.4) and window size. You can change the settings and set the figure container size before re-run the `Results`. Or you can modify the graphic window itself and save it with the menu `Save As`.

3.13 Settings

3.13.1 The population parameters estimation

K_0 is the number of burning iterations, K_1 and K_2 the numbers of SAEM iterations. If `auto` is checked, K_1 and K_2 are automatically adjusted.

The sequence of stepsizes (γ_k) decreases as $1/k^{a_j}$, $j = 1, 2$. If $a_1 = 0$ and $a_2 = 1$, then $\gamma_k = 1$ during the first K_1 iterations and $\gamma_k = 1/(k - K_1 - 1)$ during the next K_2 iterations.

l_{K_1}, l_{K_2} and r_{K_2} define the stopping rules when `auto` is checked. The number of iterations K_1 increases with l_{K_1} ; K_2 increases with l_{K_2} and decreases with r_{K_2}.

The default number of Markov chains is $L = 1$.

m_1, m_2, m_3 and m_4 are the numbers of transitions of the four different kernels used in the MCMC algorithm. The default value of m_2 is 0. Indeed this transition is recommended only in some specific cases, when the observed kinetics are very different (i.e. viral kinetics of responders, non responders, rebounders, ...).

When Simulated Annealing is checked, the temperature decreases as τ_1^k and τ_2^k. Note that you can use $\tau_j > 1$ to force the estimated variance(s) to increase from a small initial value to the estimated value. Then, $\tau_2^k > 1$ can be used if you want to obtain fits as good as possible and $\tau_1^k > 1$ if you want to obtain inter-subject variability as small as possible.

3.13.2   The individual parameters estimation
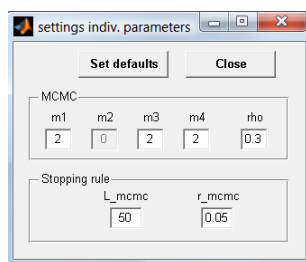
MCMC is used to estimate the conditional distributions of the individual parameters when `Estimate the cond. means and s.d.` is checked.

m_1, m_2, m_3 and m_4 are the numbers of transitions of the four different kernels used in the MCMC algorithm.

L_{mcmc} and r_{mcmc} define the stopping rule of the MCMC algorithm. The number of iterations of MCMC increases with L_{mcmc} and decreases with r_{mcmc}.

3.13.3   The log-likelihood
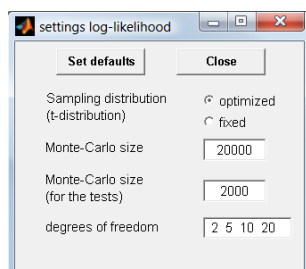
A $t-$distribution is used as proposal. The degrees of freedom number of this distribution can be either fixed or optimized. In such a case, the default possible values are 2, 5, 10 and 20 $d.f.$.

A distribution with a small number of $d.f.$ (i.e. heavy tails) should be avoided in case of stiff ODE's defined models. We recommend to set $d.f. \geq 5$.

3.13.4   The results
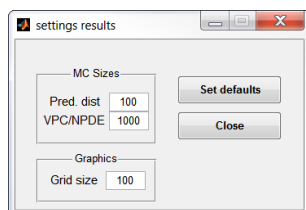
You can set the Monte-Carlo sizes used for the VPC and NPDE (default value is 500) and for Prediction distribution (default is 100). You can also set the grid size used for graphics and tables using continuous grids (like individual fits and prediction distribution).

3.13.5 Predefined scenarios
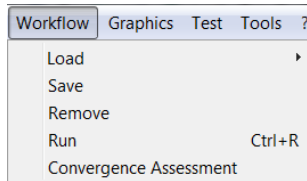
Monolix proposes the possibility to create, load and save predefined sets of settings called workflows. Each workflow include the scenario, the algorithms and results settings. Four of them are proposed but users can define new ones by selecting all the settings for the current project, and then saving it as workflow.

Chapter 4

Advanced features

The different features of MONOLIX 4.2.2 are illustrated with several examples available in the `Demos` folder.

4.1 Libraries of models

Several libraries with a large collection of pharmacokinetic and pharmacodynamic models are included in the MONOLIX software (in the folder `libraries`):

- `PK library`: 1cpt, 2cpt and 3cpts PK models; linear and nonlinear eliminations; single dose, multiple dose or steady-state designs.

- `PKe0 library`: PK models (1cpt and 2cpt) with an effect compartment (to be used with a PD model).

- `PD library`: immediate response and turnover PD models.

- `VK library`: some basic viral kinetic models (HIV and HCV).

- `Discrete library`: some basic models for count data, ordered categorical data, and repeated time to event (RTTE) models.

To use one of these models, click on the button `structural model`, or right-click on the list of currently selected models to open the `Model List` window. Select the library with the button `Monolix Library` and select one model from the list.

Of course, you can create your own libraries. If you use the MATLAB version of MONOLIX, you can write new models with MATLAB or MLXTRAN. If you use the standalone version, you should write your models with MLXTRAN. See `modelMLXTRANtutorial.pdf` for more details.

4.2   Pharmacokinetic and pharmacodynamic data

—————————————————— EXAMPLES ——————————————————

warfarin: `warfarin_PKPD1_project, warfarin_PKPD2a_project,`
`warfarin_PKPD2b_project, warfarin_PKPD3_project, warfarin_PKPD4_project`

It is possible to simultaneously analyze pharmacokinetic and pharmacodynamic data, with a pharmacokinetic function f_{PK} and a pharmacodynamic function f_{PD}. In this context,

- The observation column of the dataset contains both pharmacokinetic and pharmacodynamic observations. The `YTYPE` column indicates the type of each observation (1=PK and 2=PD),

- If you are using `.m` models of the library, the output of the f_{PK} function is an input of the function f_{PD}:

  1. Choose first the f_{PK} function and choose the corresponding error model g_{PK}.
  2. Then use the $\boxed{+}$ button to add the second model f_{PD} and choose the corresponding error model g_{PD}. It is possible to remove the selected model from the MONOLIX window using the $\boxed{-}$ button. A right click on the list, after selecting the name of the model, allows to change this model,

  Examples: `warfarin_PKPD1_project, warfarin_PKPD2a_project`

- If you write your own model using MLXTRAN, you need to define two outputs (one to fit the data defined with YTYPE=1 and one to fit the data defined with YTYPE=2).

  Examples: `warfarin_PKPD2b_project, warfarin_PKPD3_project,`
  `warfarin_PKPD4_project`

- The individual parameter vector φ is the union of the PK and PD parameters,

- Set the initial values for the fixed effects and the variance of random effects as usual, Set the initial values of the two error models,

- Run the estimation algorithm,

- The button ⊞ displays the spaghetti plots for both types of observations in two figures.

- The button ![icon] displays the chosen result figures and tables (see Section 3.8.4) for each output.

4.3   Using priors on a fixed effect

When you select to define a prior distribution on a fixed effect, a new window will open to define its law. As for individual parameters, you can choose from some given distributions (normal, log-normal, logit-normal and probit-normal) or you can define your own as a transform T of a gaussian distributed variable:

Assuming θ to be the chosen fixed effect

$$\theta = t(Z)$$

where $Z \sim \mathcal{N}(\mu_Z, \sigma_Z^2)$ is a gaussian distributed variable.

You must specify m_θ the typical value of θ ($\mu_Z = t^{-1}(m_\theta)$) and the the variance or standard deviation of Z. By default, the current initial value will be used as typical value. Also, selecting a different typical value will set it automatically as initial value for the corresponding parameter.
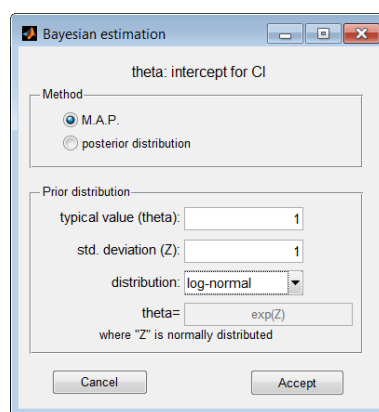
You can also choose between two estimation methods: M.A.P. and posterior distribution.

Note: MONOLIX can estimate the M.A.P only for

- gaussian priors if θ is a covariate coefficient (β)

- priors with same distribution than the corresponding individual parameter if θ is an intercept. For instance, if V is set as `log-normal` distributed, then the M.A.P of

$$\theta = \text{intercept of } V$$

can only be computed for `log-normal` priors on θ.

4.4   Categorical covariate model

———————————— EXAMPLES ————————————

Categorical covariates: PDsim1_project, PDsim2_project, PDsim3_project, phenorbabital2_project

Mixed effects models categorical covariates are described in Section A.2.3.

PDsim1_project, PDsim2_project, PDsim3_project are three pharmacodynamic examples. PDsim1_data, PDsim2_data, PDsim3_data contain simulated data obtained using an Emax model.

SEX is used as a categorical covariate: SEX= 0 for males and SEX= 1 for females.

Different models are used $C50$:

- PDsim1_data and PDsim2_data:

$$\log(C50_i) = \log(C50_{pop}) + \beta \mathbb{1}_{SEX_i=1} + \eta_{C50,i}$$

  PDsim1_data was generated using $\beta = 0$ while PDsim2_data was generated generated using $\beta = 0.4$.

- PDsim3_data:
$$\log(C50_i) = \log(C50_{pop}) + \eta_{C50,i}$$
  where $Var(C50_i) = (0.1)^2$ if $SEX_i = 0$ and $Var(C50_i) = (0.3)^2$ if $SEX_i = 1$.

phenobarbital2_project illustrates how to transform a categorical covariate.

Here, the $APGAR$ score is classified into 3 groups: Low $(APGAR <= 3)$, Medium $(4 <= APGAR <= 7)$, High $(APGAR >= 8)$. Just click on the name of the groups to rename them.
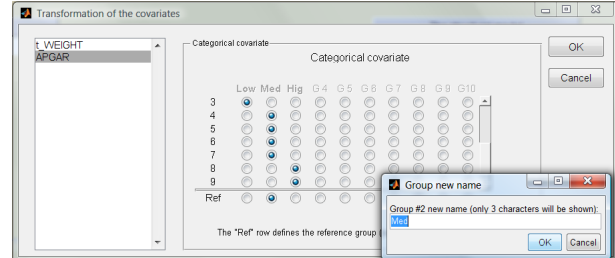
4.5   Model with censored data

4.5.1   Modeling BLQ data

―――――――――――― EXAMPLES ――――――――――――

theophylline: theophylline_cens1_project, theophylline_cens2_project

Mixed effects models with BLQ data are described Section A.5.

As an illustration, we will consider a censored version of the theophylline data.

The data files `theophylline_cens1_data.txt` and `theophylline_cens2_data.txt` are the same as the original data file `theophylline_data.txt` used in the previous section but with several left-censored observations (y_{ij}). The limit of quantification (LOQ_{ij}) can be the same for all the patients (see `theophylline_cens1_data.txt`) or it can vary from one patient to another, and even from one observation time to another (see `theophylline_cens2_data.txt`) . In the data set which includes BLQ data, the censored observations are replaced with the *LOQ* values and a new column `CENS` is added with a binary variable equal to 1 if the observation is censored and 0 otherwise.

ID	DOSE	TIME	Y	WEIGHT	CENS
1	4.02	0	.	79.6	0
1	.	0.25	3.00	79.6	1
1	.	0.57	6.57	79.6	0
⋮	⋮	⋮	⋮	⋮	⋮
2	4.4	0	.	72.4	0
2	.	0.27	2.5	72.4	1
2	.	0.52	7.91	72.4	0
⋮	⋮	⋮	⋮	⋮	⋮
12	5.3	24.15	3.00	60.5	1

In the `Data Information` window, this new column is referenced with the `CENS` header.

Then, MONOLIX can be used as usual. The button displays the same figures as the usual MCMC-SAEM with points associated to a censored observation displayed differently.

4.5.2 Modeling interval censored data

───────────────── EXAMPLES ─────────────────

censored_data: `censored_project`

──

In order to handle right-censored data, the column `CENS` can take -1 as value to tell that $y_{ij}^{(cens)} \geq LOQ_{ij}$ where LOQ_{ij} is the value given in the dataset on the corresponding line of column `Y`.

To give both limits of interval censored data, a new column `LIMIT` is needed to specify the lower limit while the `Y` value gives the upper limit. For this, the `CENS` column must be 1.

For instance, assume that we have the following dataset

(1)	ID	TIME	AMT	Y	YTYPE	LIMIT	CENS
(2)	1	0	50.00	.	.	.	.
(3)	1	0	.	90	2	.	-1
(4)	1	0.5	.	2	1	.	1
(5)	1	1	.	2	1	1	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Line (3) says that the observation of type $YTYPE = 2$ at time 0 is right-censored ($y_{11}^{(2)} \geq 90$), line (4) says that the first observation of type $YTYPE = 1$ is left-censored ($y_{11}^{(1)} \leq 2$), and line (5) says that the observation is interval censored ($y_{12}^{(1)} \in (1, 2)$) because of the existence of a numerical value in column `Limit`

4.6 Model with inter-occasion variability

——————————————— EXAMPLES ———————————————

IOV: `iov1_project, iov2_project, iov3_project, iov4_project`

Mixed effects models with IOV are described Section A.6.

Occasions are defined in the datafile, using a column `OCC` or a column `EVID`.

Then inter-occasion variabilility (IOV) can be introduced in the model with the `IOV` button. The covariate model and the covariance structure of the IOV are defined in a new window. According to the model described in Section A.6, we consider two kinds of covariates. The covariates that do not change with the occasion are displayed in the main MONOLIX window and the covariates that change with the occasion are displayed in the IOV window.

Example 1: `iov1_project`

A cross-over study was simulated. The data set `iov1a_data.txt` contains the column `OCC` and two categorical covariates, `TREAT` for treatment and `SEX`. Since the first time for each occasion is 0, MONOLIX guesses that there is no "overlapping" between the occasions.
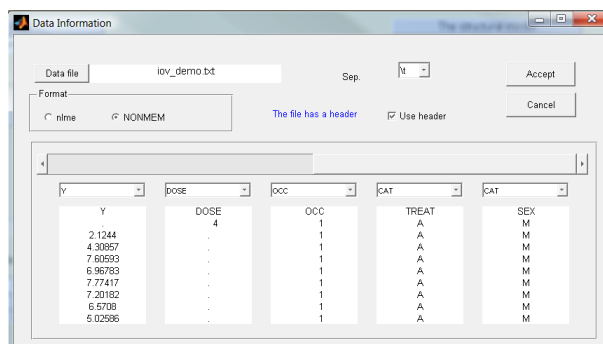
Here, the treatment (A or B) changes with the occasion. Thus, `TREAT` is a covariate that will appear in the IOV window as well as `OCC`. Furthermore, the sequence of treatments is not the same for all the subjects (A-B for some patients and B-A for the other ones). Thus, the treatment sequence (denoted `S-TREAT`) can be considered as a new covariate (this does not change with the occasion). MONOLIX automatically creates this new categorical covariate and displays it in the main MONOLIX window.

On the other hand, `S-OCC` is not created as a new categorical covariate because all the subjects have the same sequence 1-2 in this example.
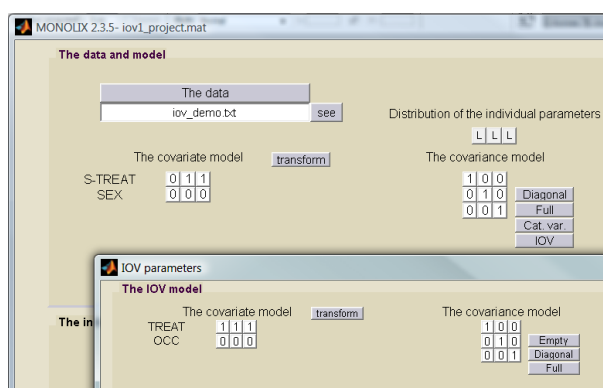
SAEM estimates the two covariance matrices and the coefficient of the different covariates:

```
Estimation of the population parameters
```

		parameter	s.e.	r.s.e.(%)	p-value
ka	:	1.51	0.083	5	
beta_ka(TREAT_B)	:	0.0225	0.042	186	0.59
V	:	0.469	0.014	3	
beta_V(TREAT_B)	:	-0.196	0.02	10	0
beta_V(S-TREAT_B-A)	:	0.0388	0.038	98	0.31
Cl	:	0.0377	0.0017	4	
beta_Cl(TREAT_B)	:	-0.195	0.031	16	3.1e-010
beta_Cl(S-TREAT_B-A)	:	0.136	0.059	44	0.022
omega_ka	:	0.277	0.039	14	
omega_V	:	0.0971	0.016	17	
omega_Cl	:	0.152	0.026	17	
gamma_ka	:	0.0794	0.035	43	
gamma_V	:	0.041	0.016	39	
gamma_Cl	:	0.118	0.017	14	
a	:	0	-	-	
b	:	0.132	0.0041	3	
c	:	1	-	-	

```
Categorical covariates
    TREAT
        Reference group: A
        Other groups: B

    S-TREAT
        Reference group: A-B
        Other groups: B-A

    SEX
        Reference group: F
        Other groups: M

    OCC
        Reference group: 1
        Other groups: 2


Elapsed time is 9.05 seconds.
CPU time is 8.95 seconds.
>>
```

Example 2: `iov2_project`

We use the same simulated data as in the previous example (`iov1_data.txt`). Here, the occasions are defined with `EVID=4` and the initial times for each occasion are arbitrary.

Example 3: `iov3_project`

Data with several occasions without wash-out was simulated. Here, the occasions are defined with the column `OCC`. Without `EVID=4` and with increasing times, MONOLIX guesses that there is some "overlapping" between the occasions

Important: Overlapping between occasions can only be handled with a model defined by a system of differential equations using MLXTRAN. Analytic solution of the PK model (*i.e.* MATLAB models of the PK library) cannot be used in this situations.

Example 4: `iov4_project`

There is an additional level of inter-occasion variability. Data was simulated with the following covariance structure:

$$
IIV = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \;,\; IOV^{(1)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \;,\; IOV^{(2)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}
$$

Running population parameter estimations, you will see that, for each variability level, the estimated variances are closed to 0 for those parameters that where simulated without variability at that levels.

Note: When there is `OCC` column or `EVID`= 4, it is not possible to use the simulation interface, nor hypothesis tests on covariance models.

4.7 Discrete data models

discrete data: `categorical1_project`, `categorical2_project`, `count1_project`, `count2_project`, `markov1a_project`, `markov1b_project`, `markov2_project`, `hmm0_project`, `hmm1_project`, `rtteExpo_project`,
warfarin: `warfarin_cat1_project`, `warfarin_cat2_project`

Mixed effects models for discrete data are described in Section A.7.

All the MLXTRAN models shown in this section are also stored in the `discreteLibrary` folder.

Either MLXTRAN or MATLAB (if you are using the MATLAB version of MONOLIX) can be used to create models for discrete data. The output of the model should be the log-likelihood.

4.7.1 ordered categorical data models

In these examples, the outcome can take four possible values: 0, 1, 2, 3.

Example 1: `categorical1_project`

Its MLXTRAN model is `categorical1_mlxt.txt`:

```
DESCRIPTION: Ordered categorical model

INPUT:
parameter = {th1, th2, th3}
regresor  = {PER, DOSE}

OBSERVATION:

level = {
  type = categorical
  categories = {0, 1, 2, 3}
  logit(P(level<=0)) = th1
  logit(P(level<=1)) = th1 + th2
  logit(P(level<=2)) = th1 + th2 + th3
}

OUTPUT:
output = level
```

$$
\begin{aligned}
P(Y = 0) &= \frac{1}{1 + e^{-\theta_1}} \\
P(Y \leq 1) &= \frac{1}{1 + e^{-\theta_1 - \theta_2}} \\
P(Y \leq 2) &= \frac{1}{1 + e^{-\theta_1 - \theta_2 - \theta_3}}
\end{aligned}
$$

Example 2: `categorical2_project`

Its MLXTRAN model is `categorical2_mlxt.txt`:

```
DESCRIPTION: Ordered categorical model with regression variables

INPUT:
parameter = {th1, th2, th3, th4, th5}
regresor = {PER, DOSE}

OBSERVATION:

level = {
  type = categorical
  categories = {0, 1, 2, 3}
  logit(P(level<=0)) = th1 + th4*PER + th5*DOSE
  logit(P(level<=1)) = th1 + th4*PER + th5*DOSE + th2
  logit(P(level<=2)) = th1 + th4*PER + th5*DOSE + th2 + th3
}

OUTPUT:
output = level
```

$$
\begin{aligned}
P(Y = 0) &= \frac{1}{1 + e^{-\theta_1 - \theta_4\, PER - \theta_5\, DOSE}} \\
P(Y \leq 1) &= \frac{1}{1 + e^{-\theta_1 - \theta_4\, PER - \theta_5\, DOSE - \theta_2}} \\
P(Y \leq 2) &= \frac{1}{1 + e^{-\theta_1 - \theta_4\, PER - \theta_5\, DOSE - \theta_2 - \theta_3}}
\end{aligned}
$$

4.7.2   count data models

In these examples, the outcome can take any positive value: 0, 1, 2, 3,....

The MLXTRAN models are, respectively `count1_mlxt.txt` and `count2_mlxt.txt`.

Example 1: `count1_project`

```
DESCRIPTION: Basic Poisson model

INPUT:
parameter = lambda

OBSERVATION:
Y = { type = count,
      log(P(Y=k)) = -lambda + k*log(lambda) - factln(k) }

OUTPUT:
output = Y
```

$$P(Y = k) \quad = \quad \frac{e^{-\lambda}\lambda^k}{k!}$$

Example 2: `count2_project`

```
DESCRIPTION: Count data model, negative binomial distribution

INPUT:
parameter = {delta, lambda}

OBSERVATION:
Y = {
  type = count

  h1 = 1/(1+lambda*delta)
  llam = log(h1)/delta
  lh2 = log(1-h1)

  lg1 = gammaln(k+1/delta)
  lg2 = gammaln(1/delta)

  if (k > 0)
    aux = llam + lg1 - lg2 + k*lh2 - factln(k)
  else
    aux = llam
  end

  log(P(Y=k)) = aux
}

OUTPUT:
output = Y
```

$$P(Y = k) \quad = \quad \frac{\Gamma(k + 1/\delta)}{k!\,\Gamma(1/\delta)}\,(1 + \delta\lambda)^{-\frac{1}{\delta}}\left(\frac{\lambda}{\lambda + 1/\delta}\right)^k$$

4.7.3   discrete Markov models

For discrete Markov models, you must specify the transition probability matrix and, optionally the probability for the initial state.

Here we show two examples of models `markov1b_mlxt.txt` and `markov2_mlxt.txt` with 2 and three states respectively.

```
INPUT:
parameter =  {p, p11, p21}

OBSERVATION:
State = {
  type = categorical
  categories = {1,2}
  dependence = Markov
  P(State_1=1)= p
  P(State=1|State_p=1) = p11
  P(State=1|State_p=2) = p21
}

OUTPUT:
output=State
```

```
INPUT:
parameter = {a11, a12, a21, a22, a31, a32}

OBSERVATION:
State = {
  type = categorical
  categories = {1,2,3}
  dependence = Markov
  logit(P(State<=1|State_p=1)) = a11
  logit(P(State<=2|State_p=1)) = a11+a12
  logit(P(State<=1|State_p=2)) = a21
  logit(P(State<=2|State_p=2)) = a21+a22
  logit(P(State<=1|State_p=3)) = a31
  logit(P(State<=2|State_p=3)) = a31+a32
}

OUTPUT:
output=State
```

4.7.4 hidden Markov models

- For $i = 1, 2, \ldots, N$, let $z_i = (z_{ij}, j \geq 1)$ be a Markov Chain with memory 1 that takes its values in $\{1, 2, \ldots M\}$:

$$P(z_{ij} = m | z_{i,j-1}, z_{i,j-2}, \ldots z_{i,1}) = P(z_{ij} = m | z_{i,j-1})$$

- Let $p_{\ell m i} = P(z_{ij} = m | z_{i,j-1} = \ell)$ and $\Pi_i = (p_{\ell m i})$ be the transition matrix of the Markov Chain z_i.

- Assume that (y_{ij}) takes discrete values in $\{0, 1, 2, \ldots\}$ and that

$$P(y_{ij} = k | z_{i1}, z_{i2}, \ldots) = P_{\psi_i}(y_{ij} = k | z_{ij})$$

Objective: Estimate the population distributions of the transition matrices (Π_i) and the individual parameters (ψ_i).

In the following examples, mixtures of $M = 2$ Poisson distribution are used. The MATLAB models `hmm1_mlx.m` and `hmm0_mlx.m` are stored in the `discreteLibrary` folder.

Example 1: `hmm1_project`

The model `hmm1_mlx.m` assumes a HMM with $M = 2$ states

Here, $M = 2$,

$$\Pi = \left(\begin{array}{cc} p_{11} & p_{12} \\ p_{21} & p_{22} \end{array} \right)$$

$$p_{11} = \frac{1}{1 + e^{-\beta_1}} \quad ; \quad p_{12} = 1 - p_{11}$$

$$p_{21} = \frac{1}{1 + e^{-\beta_1 + \beta_2}} \quad ; \quad p_{22} = 1 - p_{21}$$

and 2 Poisson distributions

$$\begin{array}{ll} \text{if } z = 1\,, & y \sim \text{Poisson}(\lambda_1) \\ \text{if } z = 2\,, & y \sim \text{Poisson}(\lambda_2) \end{array}$$

Example 2: `hmm0_project`

Here, we use the same conditional distributions for (y_{ij}) but the (z_{ij}) are sequences of *independent* random variables (Markov chain with memory 0).

$$p_1 = \frac{1}{1 + e^{-\beta}} \quad ; \quad p_2 = 1 - p_1$$

4.7.5 repeated time to event models (RTTE)

Examples: `rtteExpo_project, rtteWeibull_project`

Repeated events are observed at random times between day 0 and day 365 for 500 patients. The first event which occurs after day 365 is not observed: the time of this event is censored. We consider here a very simple model with a constant hazard function

$$h(t) = H_{base}$$

In the data file `rtte_data.txt`, we use `EVENT=1,2,...` for the observed events and `EVENT=0` for the censored event:

ID	TIME	Y
1	0	0
1	153	1
1	262	2
1	365	0
2	0	0
2	34	1
2	109	2
2	365	0

The Mlxtran model is then

```
DESCRIPTION: RTTE with constant hazard function

INPUT:
parameter = Te

OBSERVATION:
Event = {type=event, hazard=1/Te}

OUTPUT:
output = Event
```

Other hazard functions can be defined as shown in `rtteWeibull_project` demo.

Note: It is not possible to use the simulation interface for `RTTE` data.

4.7.6   joint modelling of continuous and discrete outputs

To illustrate the ability of Monolix to model jointly continuous and discrete outputs, we use the warfarin data assuming here that the PD of warfarin is an ordered categorical data with 3 possible scores: 1 if $PCA < 30$, 2 if $30 \geq PCA \leq 50$, 3 if $PCA > 50$. This new data is stored in `warfarin_cat_data.txt`.

The Mlxtran models `oral1_categorical_mlxt.txt` and `oral1_ke0_categorical_mlxt.txt` are stored in the `libraryMLXTRAN` folder next to the project file. Both models assume that the probability of a high (resp. low) PCA decreases (resp. increases) with the concentration.

Example 1: `warfarin_cat1_project`

An immediate response model is used in this example:

```
DESCRIPTION: First order oral absorption with a lag-time, and ordered categorical data

INPUT:
parameter = {Tlag, ka, V, Cl, th1, th2, th3}

PK:
Cc= pkmodel(Tlag,ka,V,Cl)

OBSERVATION:
Level = {
type=categorical
categories={1,2,3}
logit(P(Level<=1)) = -th1 + th2*Cc
logit(P(Level<=2)) = -th1 + th2*Cc + th3
```

```

output = {Cc, Level}
```

Example 2: `warfarin_cat2_project`

We use the same model as previously with an additional effect compartment

```
DESCRIPTION: First order oral absorption with a lag-time, effect compartment,
             and ordered categorical data

INPUT:
parameter = {Tlag, ka, V, Cl, ke0, th1, th2, th3}

EQUATION:
{Cc,Ce}= pkmodel(Tlag,ka,V,Cl,ke0)

OBSERVATION:
Level = {
type=categorical
categories={1,2,3}
logit(P(Level<=1)) = -th1 + th2*Ce
logit(P(Level<=2)) = -th1 + th2*Ce + th3
}

OUTPUT:
output = {Cc, Level}
```

Demos `pkrtteExpo_project` and `pkrtteWeibull_project` are two examples where PK data is modeled at the same time than RTTE data.

4.8   Complex residual error models

──────────────── EXAMPLES ────────────────

error model: `PKcorr1_project`, `PKcorr2_project`, `Emax_errorband_project`

──

Residual error models are described in Section A.3 and their use with MONOLIX is described in Section 3.3.7.

4.8.1   autocorrelated residual errors

Assuming autocorrelated residual errors is extremely easy with MONOLIX. Choose your error model as usual and just select the autocorrelation option with the checkbox:
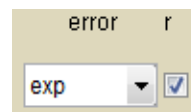
PKcorr1_project and PKcorr2_project are two PK examples where an exponential residual error model with autocorrelated errors (ε_{ij}) is assumed:

$$\log(y_{ij}) \quad = \quad \log(f(t_{ij}; \psi_i)) + \varepsilon_{ij}$$

- Equally spaced sampling times are used in PKcorr1_data: $t_{i,j+1} - t_{ij} = 1$. Then,

$$\mathrm{corr}(\varepsilon_{ij}, \varepsilon_{ij'}) = \rho^{|j'-j|}$$

- Irregular sampling times are used in PKcorr2_data. Then,

$$\mathrm{corr}(\varepsilon_{ij}, \varepsilon_{ij'}) = \rho^{|t_{ij'}-t_{ij}|}$$

Data was simulated using $\rho = 0.5$ for both examples.

4.8.2   residual errors for bounded data

Example: Emax_errorband_project

We assume in this example that the data takes continuous values in $(0, 100)$. Then, we consider the following residual error model:

$$\log\left(\frac{y_{ij}}{100 - y_{ij}}\right) = \log\left(\frac{f(t_{ij}; \psi_i)}{100 - f(t_{ij}; \psi_i)}\right) + \varepsilon_{ij}$$

The error model band(0,100) is predefined in MONOLIX and can be selected:
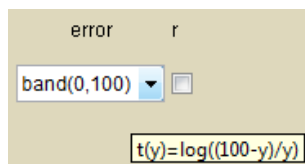
4.9 Complex PK models

PK models: `admin1_project`, `admin2_project`, `infusion_2cpt_project`, `ss1_project`, `admin2_project`, `oral0_1cpt_MD_project`, `oral0_2cpt_SS_project`, `bolus_1cptMM_project`, `phenobarbital_project`

4.9.1 Complex administrations

Example 1: `admin1_project`

The data file `admin1_data.txt` contains real PK data. This example is a combination of oral and IV bolus administrations. Then the model takes into account the bioavailability.

The MLXTRAN model `admin1_mlxt.txt` is stored in the folder `libraryMLXTRAN`.

```
DESCRIPTION: Combination of oral and IV bolus administrations.

INPUT:
parameter = {F, ka, V, CL}

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Version 1 using PK macros

PK:
compartment(amount=Ac)
absorption(adm=1, ka, p=F)
iv(adm=2)
elimination(k=CL/V)
Cc=Ac/V

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Version 2 using PK macros and ODEs

;PK:
;compartment(cmt=1, amount=Ad)
;compartment(cmt=2, amount=Ac)
;iv(adm=1, cmt=1, p=F)
;iv(adm=2, cmt=2)

;EQUATION:
;k=CL/V
;ddt_Ad = -ka*Ad
;ddt_Ac =  ka*Ad - k*Ac
;Cc=Ac/V

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

OUTPUT:
output = Cc
```

Example 2: `admin2_project`

The data file `admin2_data.txt` contains real PK data. This example is a combination of 3 oral and 1 infusion administrations. Then the model takes into account the 3 bioavailabilities.

The MLXTRAN model is `admin2_mlxt.txt`.

4.9.2 Steady-state

Example: `ss1_project`

The data file `ss1_data.txt` contains real PK data. This example is a combination of single dose and steady-state.

MLXTRAN models do not handle steady state administrations, and PK library models only handles this kind of doses, when all the subjects have exactly one steady doses. That is why, MONOLIX adds some doses artificially as an approximation of the steady state, converting the doses to the multiple doses case, but ensuring that no doses are added before a previous dose, event or observation of each subject. You can define the maximum number of doses that should be added in the preferences file (see Appendix B).

Note: Steady-state doses are treated like if there were a new occasion with `EVID`= 4. That means that it is possible to add inter-occasion variability and that it will not be possible to use the simulation interface nor the hypothesis test on covariance model.

4.10 Mixture models and model mixtures

————————————— EXAMPLES —————————————

mixtures: `PKmixture_project, bsmm_project, wsmm_project`

Mixture models and mixture of models are described in Section A.8.

4.10.1 Mixture models

Click on the `mixture` button to create latent covariates and define the number of categories for each latent covariate.
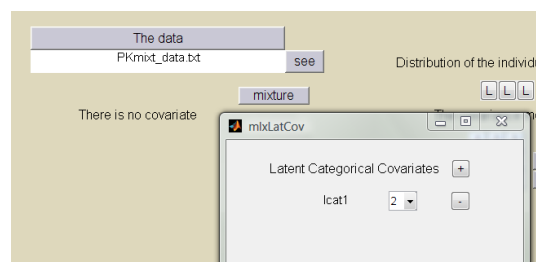
Then consider these latent covariates that you have created as a categorical covariate and define your covariate model as usual.
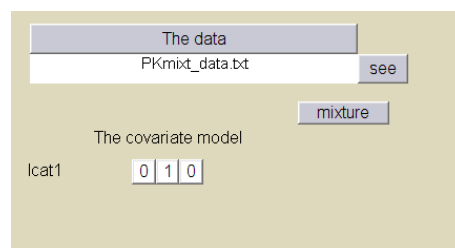
See `PKmixt_project` as an illustrative example: PK data were simulated using a one compartment model and assuming two categories for the volume ($\beta = 0.5$). In this simulated example, the column `GROUP` contains the labels (*i.e.* the groups), but the column `GROUP` is set to `IGNORE`. Project `PKgroup_project` uses the same data file and the same model, but using the known categorical covariate (setting `GROUP` to `CAT`). Then, a comparison of the results provided with these two projects can be used to validate the proposed methodology for mixtures.

Similar exercises can easily be done with the examples provided for categorical covariates in Section 4.4.

4.10.2 Model mixtures

Between subjects model mixtures (BSMM) and within subjects model mixture (WSMM) must be defined in the MLXTRAN file using the reserved key-words `BSMM` or `WSMM` (see the examples below).

On the other hand, the statistical models (defined with the Graphical User Interface) are slightly different:

- BSMM assumes no inter-individual variabilities for the proportions of each group (i.e. the probabilities to belong to the different groups)

- WSMM assumes inter-individual variabilities for the proportions of each model.

bsmm_project is an example of BSMM.

Probabilities p and $1 - p$ are associated to models f_1 and f_2. Here, p is an unknown population parameter (i.e. with no inter-individual variability).

The MLXTRAN file bsmm_mlxt.txt uses the reserved key-word BSMM.

```
DESCRIPTION: Between subject model mixture

INPUT:
parameter = {a, b, c, p}

EQUATION:
f1 = a
f2 = b*exp(-c*t)
f=bsmm(f1, p, f2, 1-p)

OUTPUT:
output = f
```

wsmm_project is an example of WSMM.

Proportions p and $1 - p$ are associated to models f_1 and f_2. Here, p is an individual parameter logit-normally distributed with some inter-individual variability (to be estimated).

The MLXTRAN file wsmm_mlxt.txt uses the reserved key-word WSMM.

```
DESCRIPTION: Within subject model mixture

INPUT:
parameter = {a, b, c, p}

EQUATION:
f1 = a
f2 = b*exp(-c*t)
f=wsmm(f1, p, f2, 1-p)

OUTPUT:
output = f
```

4.11   Tables

The ▆ button can generate some tables (see Section 3.8.2)) containing several informations: predictions, residuals,. . . .

It is possible to include several additional columns in those tables by appending a `table` keyword in section OUTPUT: at the end of the MLXTRAN script file as explained in MLXTRAN models documentation.

For each variable set in `table`, a column will be created with the values of those variables for each estimator of the individual parameters.

For instance, if you want to include the volume and the clearance used to compute the predictions, append the line

```
table={V,Cl}
```

to the section OUTPUT: of the MLXTRAN model.

4.12   Using Monolix in Matlab command line or scripts

It is provided some Matlab commands in order to access to the functionalities of Monolix from Matlab command line. It is useful when it is not desired or possible to use the software interface. It allows also to write some Matlab scripts to, for instance, estimate population parameters for several projects.

Important: There are two important things to know when using Monolix in command line

- The command line functions should not be used at the same time that the main interface (in particular methods that generate graphics). It is recommended also to close all the figures obtained for one project before creating new ones.

- Users under floating licenses must be aware that a token is reserved from first use of Monolix commands, and it must be freed manually. For that, you can close Matlab or call

  » `clear classes;`

As explained in Chapter 3, Monolix requires a project and for that we provide the class `MonolixProject`.

You can create one from any Monolix project file
  » `aMonolixProject = MonolixProject('Path/to/myProject.mlxtran');`
or you can load a new one if the object is already created
  » `aMonolixProject = MonolixProject('Path/to/myProject1.mlxtran');`

  » `...`

  » `aMonolixProject.load('Path/to/myProject2.mlxtran');`

Once the project created, the class `MonolixProject` proposes methods to:

- 1) define a scenario / workflow:
  - » `aMonolixProject.setSaem()`: to include or not population parameters estimation on the scenario. Computes also a first, rough, estimation of the conditional mean and conditional variance of the individual parameters.
  - » `aMonolixProject.setFisher()`: to define if Fisher information matrix estimation (and so standard error of estimates) should be included in the scenario and which algorithm (linearization or stochastic approximation) must be used.

- – » `aMonolixProject.setIndiv()`: to include or not individual parameter estimators and says which estimator to compute among *conditional mode* and *conditional mean* (both can be estimated at the same time). In the second case, it estimates also the conditional standard deviation and variance.

- – » `aMonolixProject.setLogLikelihood()`: to include or not the log-likelihood estimation and specify the algorithms to use between linearization and importance sampling. Both can be estimated at the same time.

- – » `aMonolixProject.setGraphics()`: to include or not the graphics and tables generation on the scenario. Allows also to say which graphic to create and/or save and which tables should be created. You can also choose the graphic format to be used for saved figures.

- 2) execute a scenario / workflow:

  - – » `aMonolixProject.run()`: runs the project scenario as explained in Section 3.7.6.

- 3) execute individual tasks:

  - – » `aMonolixProject.runSaem()`: executes only the population parameters estimation
  - – » `aMonolixProject.runFisher()`: executes only the Fisher information matrix estimation
  - – » `aMonolixProject.runIndiv()`: executes only the individual parameters estimation.
  - – » `aMonolixProject.runLogLikelihood()`: executes only the log-likelihood estimation.
  - – » `aMonolixProject.runGraphics()`: produce graphics and results tables.
  - – » `aMonolixProject.convergenceAssessment()`: executes the convergence assessment tool of the algorithms as described in Section 3.7.7 with default arguments. It can be specified the number of replicates, the parameters to simulate the initial values, and the intervals for simulation. It returns the `.mat` file holding the results.
  - – » `aMonolixProject.simulationEstimation()`: executes a project scenario on the original dataset and on several new datasets simulated using the previously estimated parameters. It returns a structure with all estimations so the users can make their own analysis. This tool can not be used with *Event* observations type, and do not simulate censored data. It allows to specify the number of replicates, the sources of variability (estimator uncertainty, individual parameter model, etc), among others. In order to reproduce the same simulated datasets, you can specify also the `seed` and/or random number generator stream (see `RandStream` help in matlab).
  - – » `aMonolixProject.runSimulation()`: allows to simulate a dataset. Several options are possible in order to choose the variability sources, the design, the population parameters, etc.

- 4) control the display:

  - » `aMonolixProject.toggleFigures();`: sets the display configuration when individual task or complete scenario is ran.
    Note: if you are working in a no-desktop environment, you should enter
    » `aMonolixProject.toggleFigures('OFF');`

- 5) manage objects:

  - » `aMonolixProject.load()`: see hereabove
  - to save the project in a new directory / file enter:
    » `aMonolixProject.save('Path/to/myNewProject.mlxtran');`
  - to overwrite the original project file:
    » `aMonolixProject.save();`

- 6) miscellaneous:

  - » `folder=aMonolixProject.getResultsFolder();`: returns the folder in which all the result files are saved (see description in Chapter 3)
  - » `aMonolixProject.setResultsFolder('Directory',newfolder);`: to define the folder where the resul files will be saved.

For further details on the `MonolixProject` class, or to see the list of available methods, type

» `doc MonolixProject`

or, » `help MonolixProject`

In order to have more information about a given method and its arguments, type

» `help MonolixProject/method`

Important: The old functions proposed to user to create its own scripts are deprecated and they will be removed in a future version. In order to adapt their scripts to Monolix evolution, the function `ver` should be used to know the Monolix version from Matlab command line:

» `v=ver('monolix');`

» `v.Version`

For older versions of Monolix, Matlab function `ver` will return empty (`[]`).

Is it also possible to know if Matlab version is compatible with the Monolix package that has been installed. For instance

» `if  mlxIsMatlabCompatible();error('Wrong Matlab version');end`

4.13   Full script projects

As said before, there are several formats to save your projects. Besides the binary `.mat` format, MONOLIX proposes a human readable `ASCII` file with `.mlxtran` extension, and an XML-like format with `.xmlx` extension.

Saving a project in the MLXTRAN format will create two `.xmlx` files with the advanced algorithm and graphics settings and the `.mlxtran` file with the description of the project (statistical model, dataset definition, structural model, etc):

```
; this script is generated automatically

DESCRIPTION:
warfarin_PK_project.mlxtran

DATA:
    path = "%MLXPROJECT%/",
    file  ="warfarin_data.txt",
    headers = {ID,TIME,DOSE,Y,YTYPE,COV,COV,CAT},
    columnDelimiter = "\t"

VARIABLES:
    age [use=cov],
    sex [use=cov,  type=cat],
    wt,
    t_wt = log(wt/70) [use=cov]

INDIVIDUAL:
    Cl = {distribution=logNormal, covariate=t_wt, iiv=yes},
    V = {distribution=logNormal, covariate=t_wt, iiv=yes},
    ka = {distribution=logNormal, iiv=yes},
    tlag = {distribution=logNormal, iiv=yes}

STRUCTURAL_MODEL:
    file = "oral1_1cpt_TlagkaVCl",
    path = "%MLXPATH%/libraries/PKLibrary",
    output = {Cc}


OBSERVATIONS:
    concentration = {type=continuous, prediction=Cc, error=combined1}
    .
    .
    .
```

In the case that the two configuration files are missing, a warning will appear when loading the project and MONOLIX will use the defaults settings. For description on how to write your own MLXTRAN projects, see `ProjectMLXTRAN.pdf`.

The XML-like format (`.xmlx` extension) is composed by only one file containing all the project information

```
<monolix>
  <project name="warfarin_PK_project.xml">
    <covariateDefinitionList>
      <covariateDefinition columnName="wt" name="t_wt" transformation="log(cov/70)" type="continuous"/>
      <covariateDefinition columnName="age" type="continuous"/>
      <covariateDefinition columnName="sex" type="categorical">
        <groupList>
          <group name="0" reference="true" values="0"/>
          <group name="1" values="1"/>
        </groupList>
      </covariateDefinition>
</covariateDefinitionList>
    <settings>
      <tasks>
        <scenario computeResults="reduced" estimateFisherInformationMatrix="true"
         estimateIndividualParameters="true" estimateLogLikelihood="false" estimatePopulationParameters="true"/>
        <individualParameterAlgorithms conditionalDistribution="false" conditionalMode="true"/>
        <logLikelihoodAlgorithms importantSampling="true" linearization="true"/>
        <fisherInformationMatrixAlgorithms linearization="true"/>
      </tasks>
      <options>
        <estimateVariances value="true"/>
        <showStandardErrorsInPercents value="true"/>
        <resultFolder uri="%MLXPROJECT%/warfarin_PK_project" value="automatic"/>
      </options>
      .
      .
      .

</project>
</monolix>
```

4.14   Preferences

There are several preferences that can be customized by each user. They are stored in `<user directory path>/monolixData/config/preference.xmlx` and are loaded at the start of MONOLIX. Some of them can be set from a dialog box. It can be opened from the 'Tools' menu:

It allows to personalize some user directories:

- log files

- structural model libraries

- module directory (i.e. the directory containing the structural built from MLXTRAN script files)

- working directory : generally the directory containing the projects
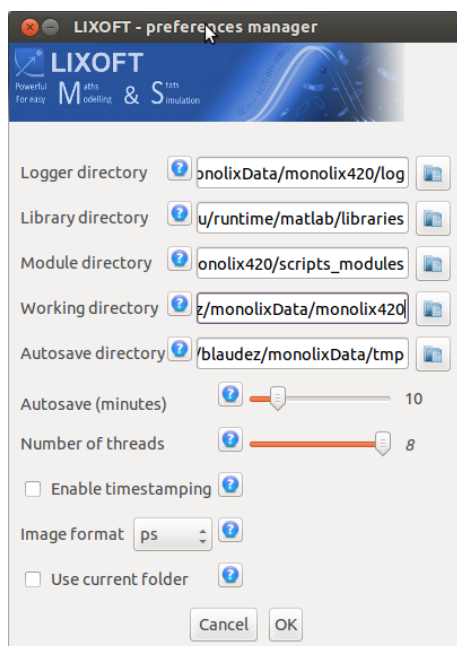
and also:

- to limit the number of threads used by MLXTRAN (structural model) - by default this maximum is the number of logical processors.

- to enable timestamping: backup of the current state of the project results folder saved in results directory as a zip file.

- to configure the saved graphics format (png, ps, jpg, bmp, tiff). Notice the 'ps' file allows to create a single file for several graphics, unlike the other formats which create one file for each graphic.

- to tell MONOLIX to use the current folder as the default folder for all the directory related dialog boxes.

Several other options are available but they must be modified directly on the file. They are divided in two categories: graphic and session related preferences. You will find more details about the content of this file in Appendix B.

For fast reference we mention here some of the session related settings that can not be modified with that dialog box.

- `historic_size`: size of the historic list for project files, and structural models.

- `editor`: text editor used to edit MLXTRAN models. It could be set also with $\boxed{\cdots}$ button in `ModelList` interface (see Section 3.3.2).

- `lockModels`: set to 1 to hide all the options related to structural model modification: options `Compile` and `Edit` on context menu on the main interface, and buttons $\boxed{\text{New}}$, $\boxed{\text{Modify}}$, $\boxed{\text{Edit}}$, $\boxed{\text{Compile}}$ on the model selection interface.

- `modelFilter`: select the default filter for the structural model: use `m`, `mlxtran` or `none` to filter on .m files, MLXTRAN files or to not filter at all.

Others settings allows to define installation preferences and are stored in the file `system.xmlx` which is located in `config` on the MONOLIX installation folder. This file is shared by all the

users and it may be needed administration rights to modify it.

They include:

- `userPath` : select the default path of the 'monolixData' directory. The path is set by default to '%USERPROFILE%' under Windows or '$HOME' under Linux. If this setting is changed, make sure to keep a user-specific path.

- `compiler` : under Linux OS, it is possible to choose the embedded compiler of MONOLIX (this setting is located in the file 'system.xmlx'). Set the argument `embed` to true, if the embedded compiler has been chosen.

- `compiler.build-linux` : under Linux OS, it is possible to configure the compiler options by using `compilation-options` tag. It is also possible to force the compiler by using `force-compiler` tag (e.g. use `icpc`, the INTEL compiler). These options have to be used carefully and may lead some problems.

- `display-license-activate` : this setting, located in the file 'system.xmlx' allows to disable the popup windows 'Lixoft Activate'

Note: Both preference files are read when MONOLIX is open, so in order to MONOLIX to take the changes into account, it must be reopened. If you use MONOLIX from MATLAB command line, then you should do

» `clear classes`

Chapter 5

PERLMLX and the batch mode

5.1   Introduction

PERLMLX is a helper tool that can be used to run

- one or several projects
- an user defined list of project files

defined through MLXTRAN or via `XMLX` project files.

PERLMLX can be executed on Windows and Linux platforms (MATLAB and standalone version of MONOLIX are supported) and it can launch multiple runs simultaneously (which can reduce drastically processing time).

All these features make PERLMLX a very efficient tool for mass processing.

PERLMLX is provided with a simple HMI but some users may prefer the command-line interface (users working on clusters will have to use this command-line interface since clusters are not managed by the HMI).
Note: PERLMLX relies on `Perl` scripts (hence its name !) which supposes of course that `Perl` has been installed on the platform.

This chapter is organized as follows:

- Section 5.2 lists the environment variables that shall be set for PERLMLX to run

- describes PERLMLX "HMI mode"

- describes PERLMLX "standalone mode"

- describes how to launch MONOLIX without PERLMLX (the so called "Batch mode")

- describes how to use PERLMLX on cluster installations.

5.2 Environment variables

When used in "HMI mode", PERLMLX requires the environment variable '`MLX_HOME`' to be set to:

- on standalone version to:
  '`/My/Path/To/Monolix.4.2.2-standalone-linux32/bin/Monolix_mcr/runtime`'

- on MATLAB version to: '`/My/Path/To/Monolix.4.2.2-standalone-linux32/runtime`'

When used in "command-line mode", it is recommended to setup environment variables (nothing mandatory but as will be seen later these variables will make the scripts easier to write):

- '`MONOLIX`' to: '`/My/Path/To/Monolix.4.2.2-standalone-linux32`'

- '`MONOLIXDATA`' to: '`/My/Path/To/monolixData/`'

These environment variables can be set with the shell commands below:

- Linux:

  - type the command lines:
    `gandalf#> export MONOLIX=/My/Path/To/Monolix.4.2.2-standalone-linux32`

  - or add the following lines in your '.bashrc' config file
    `gandalf#> gedit  /.bashrc`
    add the following line at the end of the file '.bashrc':
    `export MONOLIX=/My/Path/To/Monolix.4.2.2-standalone-linux32`

- Under Windows (XP, Vista or 7)

– type the command lines:

`c:\set MONOLIX=c:\My\Path\To\Monolix.4.2.2-standalone-win32`

– or use the graphical environment to set up the path and the variable MONOLIX to

`c:\My\Path\To\Monolix.4.2.2-standalone-win32`

Note: by default Monolix is installed beneath `c:\Document And Settings\All Users\ApplicationData` which is a hidden directory (so please refer to your operating system's documentation if Monolix directory does not show in your explorer)

5.3 HMI mode

HMI version of PerlMLX is an executable named mlxPerlScript.

The HMI is divided in eight sections:

- "Inputs control": used to define either the project files or the directories (containing the project files) on which Monolix shall be run.

  – If "Directory" is selected user can enter via the "text edit" (or with the help of the "Browse Directory" button) the directory(ies) which shall be processed. All the poject files contained in the selected directory(ies) and its sub-directories will be processed.

  – If "File" is selected user can enter via the "text edit" (or with the help of the "Browse File" button) the Mlxtran or XMLX projects that shall be processed.

  Whatever is the input selection mode, only the directories / files appended to the list (through button "Append directory / file to list") will be processed. Press button "See directories / files list" to control this list (items can be discarded from the list via "Delete element" button).

- "Output directory": directory where the results will be stored.

- "Working directory": directory where all intermediate and temporary files are stored.

- "MONOLIX directory": directory where Monolix is installed
  Something like '/My/Path/To/Monolix/matlab' for MATLAB version and '/My/Path/To/Monolix' (where 'Monolix.bat/.sh' can be found) on stand-alone installations.

- "Perl Script file": directory where the Perl script toolsRunner.pl can be found.
  Note: Button "Test PERL installation" can be used to verify that "perl" command can be accessed through the HMI. When this button is pressed a call to 'perl -v' (perl version) is done. If this call is unsuccessfull the "STD OUT / STD ERR" will be painted in yellow (perl version will be displayed in a white background otherwise). User can then use the "Set PERL directory" button to select the directory where Perl is installed.

- "MATLAB control": combo "Use MATLAB" shall be set to "NO" on standalone versions and "YES" on MATLAB verions (path to MATLAB installation shall then be entered)
  Note: path to MATLAB installation is something like '`Path/To/MATLAB/R2012a`'. What is important is tha a subdirectory '`bin`' where '`matlab.exe`' can be found exists under this path.

- "Threads control": enter the number of threads that can be opened. On multiprocessor machines this control can be used to force PERLMLX to use the maximum processing power (or on the contrary limited resources !).

- "Launcher": see below.
  Note: it is recommanded to try the perl installation before pressing this button (see "Perl Script file" item hereabove).
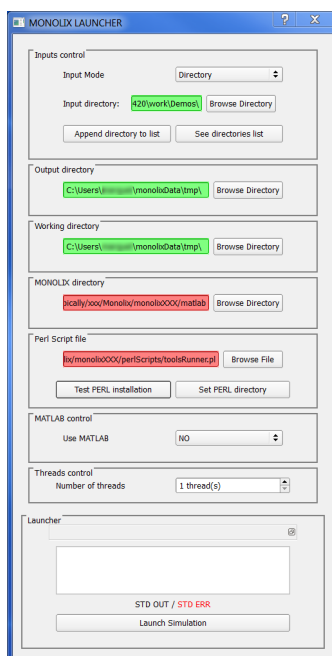
Note: if environment variable '`MLX_HOME`' has been set properly MONOLIX directory and path to the `Perl` script `toolsRunner.pl` will be automatically set. Since it is possible to detect the installation's type (standalone / MATLAB) through this variable, MATLAB combo will also be automatically set.

Note: it is possible to save and restore configuration via File menu, actions "Export / import mlxPerlScript Cfg"

Note: light green and light red are used to distinguish between valid and invalid directories

/ files names.

User shall press the "Launch Simulation" button (in the "Launcher" group box at the bottom of the HMI) to start the simulation. This group box can be docked and will show in the text edit:

- exact transcript of the launched `Perl` process

- standard output and error from the launched `Perl` process

5.4   Standalone mode

To launch PERLMLX in standalone mode, user shall use script '`toolsRunner.pl`'. Several options are made available to control this script (these options are introduced in the following sub section). It is also possible to control these options via a reusable configuration file (which is introduced in the second subsection).

5.4.1   Options

Here are the different options available for script '`toolsRunner.pl`':

argument	value	description
–tool	execute	name of tools to execute (required parameter)
–toolhelp		display help documentation
–output-directory	<directory>	output directory path, where the results are stored
–thread	<number of thread>	number of simultaneous projects to run
–input-directories	<dir1,..dirN>	list of input directories containing project files
–input-project-list	<project1, ...,projectN>	input list of projects separated by comma
–input-project-file-list	<project file list>	input file containing list of project files
–use-matlab		MATLAB version of MONOLIX instead of standalone
–matlab-path	<matlab path>	MATLAB path (if use-matlab is set)
–monolix-path	<path of monolix>	MONOLIX path
–working-directory	<directory>	directory where all intermediate and temporary files are stored
–config	<path of configuration file>	config file path

In the following paragraphs we give some examples of correct instruction sets :

Running Monolix on a demo file

```
  gandalf#> perl $MONOLIX/perlScripts/toolsRunner.pl \
--tool=execute \
--input-project-list=$MONOLIXDATA/monolix422/work/Demos/categorical_covariate/PDsim1_project.mlxtran \
--output-directory=$MONOLIXDATA/monolix422/work/output \
--monolix-path=$MONOLIX/bin \
--working-directory=/home/gandalf/tmp
```

In this example a standalone version of Monolix is launched (with the parameter '`-monolix-path`') using the Mlxtran project '`PDsim1_script.mlxtran`'. The output directory has been set to '`/home/gandalf/monolixData/monolix422/work/output`'

Running Monolix on the `Demos` directory

```
  gandalf#> perl $MONOLIX/perlScripts/toolsRunner.pl \
--tool=execute \
--input-directories=$MONOLIXDATA/monolix422/work/Demos \
--monolix-path=$MONOLIX/bin \
--output-directory=$MONOLIXDATA/monolix422/work/output \
--working-directory=/home/gandalf/tmp
```

This illustrates an execution of Monolix on all `XMLX` and Mlxtran projects present in the directory '`$MONOLIXDATA/monolix422/work/Demos`'

Running Monolix on a project file list

First edit a file list using a simple text editor (gedit, emacs, vim, kwrite, . . .) :
`gandalf#> gedit myslist.lst` then execute '`toolsRunner.pl`' :
```
gandalf#> perl $MONOLIX/perlScripts/toolsRunner.pl \
--tool=execute \
--input-project-file-list=./mylist.lst \
--output-directory=$MONOLIXDATA/monolix422/work/output \
--monolix-path=$MONOLIX/bin \
--working-directory=/home/gandalf/tmp
```

5.4.2 Setting up the configuration file

Script 'toolsRunner.pl' can also be used with 'config' option and a '.ini' configuration file (the file can be created using a simple text editor). Here is the information that this configuration file shall contain:

subsection	parameters	description
[path]	MATLAB	MATLAB path (required for a MATLAB version of MONOLIX)
	MONOLIX	MONOLIX path
[monolix]	standalone	standalone version?
[program-generic-options]	thread	number of threads: it is recommended that the number of threads is smaller or equal to the number of processor cores available on the computer
[program-execute-options]	workingDirectory	directory used by PERLMLX to store intermediate results
	command prefix	add a command prefix to a MONOLIX command, this parameter is useful in cluster mode (ex: use of 'qsub' before the MONOLIX command)
	script-embed	MONOLIX generates a script containing a command line; this parameter is used in the cluster mode: the qsub command is run on a script without any command line argument, with the command line embedded into a script
	useProjectDirectoryToSaveResult	the default directory of results is stored in same directory as project directory

In the following paragaphs examples of correct configuration files are given:

- **example using the MATLAB version under Linux OS**
  [path]
  ; MATLAB version of MONOLIX

; MATLAB path
matlab=/opt/matlab/
; directory of MONOLIX
monolix=/opt/Monolix-4.2.2-matlab-linux32
[monolix]
; standalone version? No
standalone=false
[program-generic-options]
; execution on the nodes
thread=2
[program-execute-options]
; PERLMLX need a temporary directory to store intermediate files
workingDirectory=/tmp

- **example using the standalone version under Linux OS**
  [path]
  ; MONOLIX directory
  monolix=/opt/Monolix-4.2.2-standalone-linux32/bin
  [monolix]
  ; use of the standalone version
  standalone=true
  [program-generic-options]
  ; execution on the nodes
  thread=2
  [program-execute-options]
  ; PERLMLX need a temporary directory to store intermediate files
  workingDirectory=/tmp

- **example using the MATLAB version on a Linux cluster**
  [path]
  ; MATLAB version of MONOLIX
  ; MATLAB path
  matlab=/opt/matlab/
  ; directory of MONOLIX
  monolix=/opt/Monolix-4.2.2-standalone-linux32
  [monolix]
  ; not the standalone version
  standalone=false
  [program-generic-options]
  ; on a cluster multi-threading is not necessary:
  ; the cluster uses a queue to dispatch program
  ; execution on the nodes

thread=1
[program-execute-options]
; PERLMLX need a temporary directory to store intermediate files
workingDirectory=/tmp

; Specific options to run MONOLIX on a cluster
; qsub is generally the program to submit a process on a cluster
; qsub is typically applied to a script, script-embed option
; allows to create a shell script containing the MONOLIX command
command-prefix=qsub -V -u gandalf
script-embed=true

Then, a user working on a cluster can run MONOLIX as usual:

```
gandalf#> perl /opt/Monolix-matlab/perlScripts/toolsRunner.pl \
--tool=execute \
--config=$MONOLIXDATA/work/myconfig.ini \
--input-directories=$MONOLIXDATA/monolix422/work/Demos \
--output-directory=$MONOLIXDATA/monolix422/work/output
```


Running MONOLIX on a demo file

```
  gandalf#> perl $MONOLIX/perlScripts/toolsRunner.pl \
--tool=execute \
--input-project-list=$MONOLIXDATA/monolix422/work/Demos/categorical_covariate/PDsim1_project.mlxtran \
--output-directory=$MONOLIXDATA/monolix422/work/output \
--config=$MONOLIXDATA/work/myconfig.ini
```


Running MONOLIX on the Demos directory

```
  gandalf#> perl $MONOLIX/perlScripts/toolsRunner.pl \
--tool=execute \
--input-directories=$MONOLIXDATA/monolix422/work/Demos \
--output-directory=$MONOLIXDATA/monolix422/work/output \
--config=$MONOLIXDATA/work/myconfig.ini
```

Running Monolix **on a project file list**

```
gandalf#> perl $MONOLIX/perlScripts/toolsRunner.pl --tool=execute \
--input-project-file-list=./mylist.lst \
--output-directory=$MONOLIXDATA/monolix422/work/output \
--config=$MONOLIXDATA/work/myconfig.ini
```

5.5   Batch mode in depth

5.5.1   Running Monolix without PerlMLX

It is possible to run Monolix using a simple command line:

- with the standalone version of Monolix

```
$MONOLIX/bin/Monolix.sh [-nowin] \
                        -p <project> \
                        -f [run|saem|fim|ll|graphics]
```

- with the Matlab version of Monolix (the command has to be launched from the `matlab` directory of the installation of Monolix , i.e. `<monolix install path>/matlab`)

```
matlab -wait  \
       -nosplash \
       -nodesktop \
       -r "monolix('-nowin', \
                   '-p','<project>', \
                   '-f','[run|saem|fim|ll|graphics]', \
                   '-destroy' \
                 ),exit"
```

5.5.2   Monolix **Program options**

- **-nowin** : without opening a window, mandatory in no-desktop environments.

- **-p <project>** : project to run

- **-f run|saem|fim|ll|graphics**

  - **run**: run a workflow

– **saem**: estimate population parameters

– **fim**: estimate the standard errors of the estimates and Fisher information matrix

– **ll**: estimate the log-likelihood

– **graphics**: generate the result graphics

5.5.3 Example

Run the standalone version of Monolix on the project `PDsim1_project.mlxtran`:

```
$MONOLIX/bin/Monolix.sh \
-nowin \
-p $MONOLIXDATA/monolix422/work/Demos/categorical_covariates/PDsim1_script.mlxtran -f run
```

Here Monolix is executed without displaying a window (option -nowin).

Example: shell script to run Monolix on all Mlxtran files on a directory:

```
1  if   [ -d $1 ]; then
2  for mlxtran\_i in 'find $1 -type f | grep mlxtran$'; do
3          echo "Run Monolix in $mlxtran_i";
4          $MONOLIX/bin/Monolix.sh -nowin -p $mlxtran_i -f run
5  end
6  else
7  echo "$1 is not a directory"
8  fi
```

5.6 Monolix **on cluster**

5.6.1 Cluster filesystem

To run Monolix on a cluster, each cluster node must have access to the Monolix directory and to the user home directory.

Each node shares a common file system containing the user directories. The common filesystem may also contain Matlab and Monolix . However, in the case of the standalone version of Monolix , it is recommended to have Monolix installed on a local drive for each node, since the standalone binary is large: if installed on the network, the startup time on a cluster may take too long.

5.6.2 Task submission mechanism

Generally, a task is submitted to the cluster using a specific command, e.g. `qsub` in the case of *Torque*, *PBS* or *GridEngine* (former *SGE*). This command runs a script, provided as parameter, on a cluster node choosen by the cluster scheduler.

The MONOLIX batch tool allows to run this command through the configuration file. To enable the cluster functionnality, the options `command-prefix` and `script-embed` must be set in the configuration file, where `command-prefix` is the command used to submit a task (generally `qsub`) and `script-embed` allows to encapsulate the low level MONOLIX command (see Section 5.5.1).

At this point, when the command `toolsRunner.pl` is executed, each instance of MONOLIX takes place on a cluster node chosen by the cluster scheduler.

5.6.3 Example

Setting up a configuration file

The following configuration file enables a cluster mode (e.g. *Torque, PBS, GridEngine*). A task is submitted to the cluster using the command 'qsub' which runs a script on a cluster mode.

```
[path]
; use a MATLAB version of MONOLIX
; MATLAB path
matlab=/opt/matlab/
; MONOLIX directory
monolix=/opt/Monolix-4.2.2-standalone-linux32
[monolix]
;do not use standalone version
standalone=false
[program-generic-options]
; one thread: on a cluster multi-threading is not necessary:
; the cluster uses a queue to dispatch program
; execution on the nodes
thread=1
[program-execute-options]
; PERLMLX needs a temporary directory to store intermediate files
workingDirectory=/tmp
; Specifics options to run MONOLIX on a cluster
; qsub is generally the program to submit a process on a cluster
; Typically qsub uses a script as the process to run; the script-embed option
; allows to create a shell script containing the MONOLIX command
command-prefix=qsub -V -u gandalf
script-embed=true
```

Submit tasks on a cluster

Here each project stored in the directory `$MONOLIXDATA/monolix422/work/Demos` is run on the cluster nodes. This directory has to be shared by all nodes (typically, under Linux OS, with `NFS`), with exactly the same path.

```
gandalf#> perl $MONOLIX/perlScripts/toolsRunner.pl -tool=execute \

-input-directories=$MONOLIXDATA/monolix422/work/Demos \

-output-directory=$MONOLIXDATA/monolix422/work/output \

-config=$MONOLIXDATA/work/myconfig.ini
```

where `MONOLIX` and `MONOLIXDATA` are environment variables defined as explained in Section 5.2.

Chapter 6

Validation suite

6.1 Introduction

The validation suite is the highest level of the tests, and is provided on request. It consists in a set of project runs, whose results are compared with references results. A customer can also add his own projects in the suite. A more detailed documentation on the testing strategy and the development processes is also available on request.

The process of the validation suite consists in taking a set of projects with predefined scenarii and to launch them with MONOLIX . The result files are then compared with the reference files. Due to the Matlab kernel, slight numerical differences can appear between two Matlab versions for a same project. This is why the validation suite is launched for each operating system and each Matlab version with different references. It must be executed in batch mode, i.e. via a command line, without a graphical interface.

6.2 Prerequisites

The informations below are required to run the validation suite in batch mode:

- directory path of MONOLIX,

- directory path of MATLAB (does not apply to standalone MONOLIX)

- directory path of references files

A `Perl` installation is also required to run the validation suite.

6.3   Combinations

As described above, several sets of references results are required to cover platform variabilities. The combinations are listed as follows.

MONOLIX **version**	**Matlab version**	**Operating system**
Install.exe	2008b	Windows 32 bits
R2010a-Install.exe	2010a	Windows 32 bits
R2010b-Install.exe	2010bSP1	Windows 32 bits
R2010b-Install.exe	2011a	Windows 32 bits
R2010b-Install.exe	2011b	Windows 32 bits
R2010b-Install.exe	2012a	Windows 32 bits
standalone2008b-linux32.sh	2008b	Linux 32 bits
matlab2010a-linux32.sh	2010a	Linux 32 bits
matlab2010bSP1-linux32.sh	2010bSP1	Linux 32 bits
matlab2010bSP1-linux32.sh	2011a	Linux 32 bits
matlab2010bSP1-linux32.sh	2011b	Linux 32 bits
matlab2010bSP1-linux32.sh	2012a	Linux 32 bits
standalone2008b-linux64.sh	2008b	Linux 64 bits
matlab2010a-linux64.sh	2010a	Linux 64 bits
matlab2010bSP1-linux64.sh	2010bSP1	Linux 64 bits
matlab2010bSP1-linux64.sh	2011a	Linux 64 bits
matlab2010bSP1-linux64.sh	2011b	Linux 64 bits
matlab2010bSP1-linux64.sh	2012a	Linux 64 bits

6.4   Extensive coverage through the demo projects

The projects from the validation suite are designed to provide maximum coverage of the set of functionalities of MONOLIX. The two types of structural models, Matlab and MLXTRAN, are included in these projects. Currently, the suite includes 17 projects:

- **Emax_errorband_project** : transformed error model,

- **iov3_project, ss2_project** : inter occasion variability (IOV) with several levels,

- **theophylline2_project, warfarin_PKPD1_project, warfarin_PKPD4_project, warfarin_PK_project** : methodologic cases with classical database,

- **categorical1_project, count1_project** : discrete observed variables,

- **pkrtteExpo_project, rtteWeibullCount_project** : repeated time to event,

- **PKmixt_project** : model of mixtures and mixture models,

- **PDsim1_project, phenobarbital2_project** : categorical covariates,

- **bolus_1cptMM_project, demo_project, sequential_oral0_oral1_project** : advanced and complex administrations.

6.5 Execution

Program parameters

argument	value	description
–help		display help documentation
–output-directory	\<directory\>	output directory path, where the results are stored
–thread	\<number of thread\>	number of simultaneous projects to run
–reference-directory	\<directory\>	reference directory containing project files
–use-matlab	true ou **false**	MATLAB version of MONOLIX instead of standalone
–matlab-path	\<matlab path\>	MATLAB path (if use-matlab is set)
–monolix-path	\<path of monolix\>	MONOLIX path
–keep-output	true ou **false**	saves output directory

Examples

Here we set the environment variable

- 'MONOLIXSTD' to '/opt/Monolix.4.2.2-standalone-linux32',

- 'MONOLIX' to '/opt/Monolix.4.2.2-matlab2010bSP1-linux32',

- 'MATLAB' to '/opt/MATLAB/R2010b' and

- 'VALIDATIONSUITE' to '/home/gandalf/validationSuite'

To set environment variables, see Section 5.2.

Running the validation suite with standalone version

Go to the validation suite directory (here /home/gandalf/validationSuite/scripts).
Then,

```
gandalf#> perl validationSuite.pl  \
--reference-directory=$VALIDATIONSUITE/reference/lin32/2008b \
--monolix-path=$MONOLIXSTD/bin \
--output-directory=/home/gandalf/output
```

This illustrates an execution of Monolix on all XMLX and Mlxtran projects present in the directory '$VALIDATIONSUITE/reference/lin32/2008b'. If the validation is successful, the user should get

```
[2012/04/27 16:53:00] [INFO]: Comparison results
Check ss2_project project :
 OK

 Check pkrtteExpo_project project :
 OK

 Check rtteWeibullCount_project project :
 OK

 Check warfarin_PKPD4_project project :
 OK

 Check sequential_oral0_oral1_project project :
 OK

 Check theophylline2_project project :
 OK

 Check warfarin_PKPD1_project project :
 OK

 Check warfarin_PK_project project :
 OK

 Check Emax_errorband_project project :
 OK

 Check PKmixt_project project :
 OK

 Check PDsim1_project project :
 OK

 Check categorical1_project project :
 OK

 Check demo_project project :
 OK
```

Running the validation suite with matlab version

Go to the validation suite directory (here /home/gandalf/validationSuite/scripts). Then,

```
gandalf#> perl validationSuite.pl --use-matlab=true \
--reference-directory=$VALIDATIONSUITE/reference/lin32/2010b \
--monolix-path=$MONOLIX/matlab \
--output-directory=/home/gandalf/output
--matlab-path=$MATLAB
```

Running the validation suite with matlab version and saving output directory

In the validation fails, it can be useful to store the output directory using the option 'keep-output':

```
gandalf#> perl validationSuite.pl --use-matlab=true --keep-output=true \
--reference-directory=$VALIDATIONSUITE/reference/lin32/2010b \
--monolix-path=$MONOLIX/matlab \
--output-directory=/home/gandalf/output
--matlab-path=$MATLAB
```

```
[2012/04/27 18:20:10] [INFO]: Comparison results
Check ss2_project project :
 OK

Check pkrtteExpo_project project :
 KO
  Fields with differences:
 IF_sa
 se_fixed_sa (error: 0.01)
 pv_fixed_sa (rel error: 0.01)
 se_random_sa (error: 0.01)
 se_gabc_sa (rel error: 0.01)
 CO_sa (error: 0.001)
 Cth_sa (error: 0.001)

Check rtteWeibullCount_project project :
 KO
  Fields with differences:
 IF_sa
 se_fixed_sa (error: 1)
 pv_fixed_sa (error: 0.1)
 se_random_sa (error: 0.1)
 CO_sa (error: 0.01)
 Cth_sa (error: 10)

Check warfarin_PKPD4_project project :
 OK

Check sequential_oral0_oral1_project project :
 OK

Check theophylline2_project project :
 OK

Check warfarin_PKPD1_project project :
 OK
```

The display shows the fields in the results which are not equal to reference results, and indicates the order of errors.

It exists two kinds of error:

- absolute error

```
se_fixed_sa (error: 0.01)
```

- relative error

```
pv_fixed_sa (rel error: 0.01)
```

During the comparison, the absolute error is used. In the case where this error is too large ($> 10^{-3}$), the relative error is used.

Appendix A

The statistical models

A.1   The nonlinear mixed effects model

Detailed and complete presentations of the nonlinear mixed effects model can be found in [5, 6, 21]. See also the many references therein.

We consider the following general nonlinear mixed effects model for continuous outputs:

$$y_{ij} = f(x_{ij}, \psi_i) + g(x_{ij}, \psi_i, \xi)\varepsilon_{ij} \quad , \ 1 \leq i \leq N \quad , \quad 1 \leq j \leq n_i \tag{A.1}$$

Here,

- $y_{ij} \in \mathbb{R}$ is the jth observation of subject i,

- N is the number of subjects,

- n_i is the number of observations of subject i,

- the regression variables, or design variables, (x_{ij}) are assumed to be known, $x_{ij} \in \mathbb{R}^{n_x}$,

- for subject i, the vector $\psi_i = (\psi_{i,\ell}; 1 \leq \ell \leq n_\psi) \in \mathbb{R}^{n_\psi}$ is a vector of n_ψ individual parameters:
$$\psi_i = H(\mu, c_i, \eta_i) \tag{A.2}$$
   where

   - $c_i = (c_{im}; 1 \leq m \leq M)$ is a known vector of M covariates,
   - μ is an unknown vector of fixed effects of size n_μ,

– η_i is an unknown vector of normally distributed random effects of size n_η:

$$\eta_i \sim_{i.i.d.} \mathcal{N}(0, \Omega)$$

- the residual errors (ε_{ij}) are random variables with mean zero and variance 1,

- the residual error model is defined by the function g and some parameters ξ.

Here, the parameters of the model are $\theta = (\mu, \Omega, \xi)$. We will denote $\ell(y; \theta)$ the likelihood of the observations $y = (y_{ij} \,; 1 \leq i \leq n \,, 1 \leq j \leq n_i)$ and $p(y, \psi; \theta)$ the likelihood of the complete data $(y, \psi) = (y_{ij}, \psi_i \,; 1 \leq i \leq n \,, 1 \leq j \leq n_i)$. Thus,

$$\ell(y; \theta) = \int p(y, \psi; \theta) \; d\psi.$$

Let us see now the statistical model used in MONOLIX 4.2.2 more in details.

A.2 The statistical model for the individual parameters

In MONOLIX 4.2.2 , we assume that ψ_i is a transformation of a Gaussian random vector φ_i:

$$\psi_i = h(\varphi_i) \tag{A.3}$$

where, by rearranging the covariates (c_{im}) into a matrix C_i, φ_i can be written as

$$\varphi_i = C_i \mu + \eta_i \tag{A.4}$$

A.2.1 Examples of transformations

Here, different transformations (h_ℓ) can be used for the different components of $\psi_i = (\psi_{i,\ell})$ where $\psi_{i,\ell} = h_\ell(\varphi_{i,\ell})$ for $\ell = 1, 2, \ldots, \ell$. Let us denote by $\Phi(u)$ the cumulative distribution function of a Gaussian distributed random variable.

- $\psi_{i,\ell}$ has a log-normal distribution if $h_\ell(u) = e^u$,

- assuming that $\psi_{i,\ell}$ takes its values in $(0, 1)$, we can use a logit transformation $h_\ell(u) = 1/(1 + e^{-u})$, or a probit transformation $h_\ell(u) = \Phi(u)$.

- assuming that $\psi_{i,\ell}$ takes its values in (A, B), we can define $h_\ell(u) = A + (B - A)/(1 + e^{-u})$, or $h_\ell(u) = A + (B - A)\Phi(u)$.

In the following, we will use either the parameters ψ_i or the Gaussian transformed parameters $\varphi_i = h^{-1}(\psi_i)$.

The model can address continuous and/or categorical covariates.

A.2.2 Example of continuous covariate model

Consider a PK model that depends on volume and clearance and consider the following covariate model for these two parameters:

$$CL_i = CL_{\text{pop}} \left(\frac{W_i}{W_{\text{pop}}} \right)^{\beta_{CL,W}} \left(\frac{A_i}{A_{\text{pop}}} \right)^{\beta_{CL,A}} e^{\eta_{i,1}}$$

$$V_i = V_{\text{pop}} \left(\frac{W_i}{W_{\text{pop}}} \right)^{\beta_{V,W}} e^{\eta_{i,2}}$$

Where W_i and A_i are the weight and the age of subjet i and where W_{pop} and A_{pop} are some "typical" values of these two covariates in the population. Here, ψ_i will denote the PK parameters (clearance and volume) of subject i and φ_i its log-clearance and log-volume. Let

$$W_i^\star = \log \left(\frac{W_i}{W_{\text{pop}}} \right) \quad ; \quad A_i^\star = \log \left(\frac{A_i}{A_{\text{pop}}} \right)$$

Then,

$$\varphi_i = \begin{pmatrix} \log(CL_i) \\ \log(V_i) \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 & W_i^\star & A_i^\star & 0 \\ 0 & 1 & 0 & 0 & W_i^\star \end{pmatrix} \begin{pmatrix} \log(CL_{\text{pop}}) \\ \log(V_{\text{pop}}) \\ \beta_{CL,W} \\ \beta_{CL,A} \\ \beta_{V,W} \end{pmatrix} + \begin{pmatrix} \eta_{i,1} \\ \eta_{i,2} \end{pmatrix}$$

$$= C_i \mu + \eta_i$$

A.2.3 Example of categorical covariate model

Assume that some categorical covariate G_i takes the values 1, 2, ..., K. Assume that if patient i belongs to group k, *i.e.* $G_i = k$, then

$$\log(CL_i) = \log(CL_{\text{pop},k}) + \eta_i$$

where $CL_{\text{pop},k}$ is the population clearance in group k.

Let $k^\star$ be the reference group. Then, for any group k, we will decompose the population clearance $CL_{\mathrm{pop},k}$ as

$$\log(CL_{\mathrm{pop},k}) = \log(CL_{\mathrm{pop},k^\star}) + \beta_k$$

where $\beta_{k^\star} = 0$.

The variance of the random effects can also depend on this categorical covariate:

$$\eta_i \sim \mathcal{N}(0, \Omega_k) \quad \text{if} \quad G_i = k$$

Remark: It is assumed in MONOLIX 4.2.2 that the correlation matrix of the random effect is the same for all the groups. In other words, only the variances of the random effects can differ from one group to another.

──────────────── MONOLIX ────────────────

```
Choice of the transformation is described in Section 3.3.5.
Selection of the covariate model is described in Section 3.3.3.
Examples with categorical covariates are given in Section 4.4.
```

A.3   The residual error model

The within-group errors (ε_{ij}) are supposed to be Gaussian random variables with mean zero and variance 1. Furthermore, we suppose that the ε_{ij} and the η_i are mutually independent.

Different error models can be used in MONOLIX 4.2.2 :

- the constant error model assumes that $g = a$ and $\xi = a$,

- the proportional error model assumes that $g = b\,f$ and $\xi = b$,

- a combined error model assumes that $g = a + b\,f$ and $\xi = (a, b)$,

- an alternative combined error model assumes that $g = \sqrt{a^2 + b^2\,f^2}$ and $\xi = (a, b)$,

- a combined error model with power assumes that $g = a + b\,f^c$ and $\xi = (a, b, c)$,

- ...

Furthermore, all these error models can be applied to some transformation of the data:

$$t(y_{ij}) = t(f(x_{ij}, \psi_i)) + g(x_{ij}, \psi_i, \xi)\varepsilon_{ij} \tag{A.5}$$

For example:

- the exponential error model assumes that $y > 0$:

$$
\begin{aligned}
t(y) &= \log(y) \\
y &= f e^{g\varepsilon}
\end{aligned}
$$

- the logit error model assumes that $0 < y < 1$:

$$
\begin{aligned}
t(y) &= \log(y/(1-y)) \\
y &= \frac{f}{f + (1-f)e^{-g\varepsilon}}
\end{aligned}
$$

- the logit error model can be extended if we assume that $A < y < B$:

$$
\begin{aligned}
t(y) &= \log((y-A)/(B-y)) \\
y &= A + (B-A)\frac{f-A}{f-A+(B-f)e^{-g\varepsilon}}
\end{aligned}
$$

It is possible with MONOLIX to assume that the residual errors (ε_{ij}) are correlated:

$$
\text{corr}(\varepsilon_{i,j}, \varepsilon_{i,j+1}) = \rho^{(x_{i,j+1}-x_{i,j})} \tag{A.6}
$$

Here, we assume that $0 \leq \rho < 1$ and that for any i, $(x_{i,j}, 1 \leq j \leq n_i)$ is an increasing sequence of regression scalar variables.

_____ MONOLIX _____

```
Selection of the residual error model is described Section 3.3.7.
Several examples of residual error models are provided with the demos:
```

- `combined error model:  warfarin/warfarin_PK_project`

- `exponential error model:  PK/Bolus1cptMM_project`

- `extended logit error model:  error model/Imax_errorband_project`

- `autocorrelated residual errors:  error model/infusion_correrror_project`

.

A.4   Multi-responses model

The basic model can be extended to multi-responses:

$$
\begin{aligned}
y_{ij}^{(1)} &= f_1(x_{ij}^{(1)}, \psi_i) + g_1(x_{ij}^{(1)}, \psi_i; \xi_1)\varepsilon_{ij}^{(1)} \quad , \ 1 \leq i \leq N \ , \quad 1 \leq j \leq n_{i1} \\
&\vdots \qquad \vdots \\
y_{ij}^{(L)} &= f_L(x_{ij}^{(L)}, \psi_i) + g_L(x_{ij}^{(L)}, \psi_i; \xi_L)\varepsilon_{ij}^{(L)} \quad , \ 1 \leq i \leq N \ , \quad 1 \leq j \leq n_{iL}
\end{aligned}
$$

This is useful, for example, for PKPD models in which the input of the PD model $x_{ij}^{(2)}$ is the concentration, that is the output of the PK model $f_1(x_{ij}^{(1)}, \psi_i)$.

—————————————————— MONOLIX ——————————————————

```
How to handle models with multiple outputs is described in Section 4.2.
Several examples are provided using the warfarin data:
warfarin_PKPD1_project, warfarin_PKPD2_project, warfarin_PKPD3_project,
warfarin_PKPD4_project.
```

A.5 Model with censored data

A.5.1 BLQ data

In some context, because of assay limitation, when data y_{ij} are inferior to a limit of quantification (LOQ), we do not observe y_{ij} but only the censored value LOQ. These data are usually named BLQ (Below the Limit of Quantification) data or left-censored data.

Let denote $I_{obs} = \{(i,j)|y_{ij} \geq LOQ\}$ and $I_{cens} = \{(i,j)|y_{ij} \leq LOQ\}$ the index sets of the uncensored and censored observations respectively. For $(i,j) \in I_{cens}$, let $y_{ij}^{cens} = y_{ij}$ denote the unknown value of the censored observation j of subject i. Let denote y_i^{cens} the vector of censored observations of subject i. Finally, we observe

$$y_{ij}^{obs} = \left\{ \begin{array}{lll} y_{ij} & \text{if} & (i,j) \in I_{obs}, \\ LOQ & \text{if} & (i,j) \in I_{cens}. \end{array} \right.$$

We denote $y_i^{obs} = (y_{i1}^{obs}, \ldots, y_{in_i}^{obs})$ as the observations of subject i and $y^{obs} = (y_1^{obs}, \ldots, y_N^{obs})$ the total observations dataset.

A.5.2 Interval censored data

It is possible now also to model interval censored data, i.e data where it is only known that y_{ij} is above a limit of detection LOD_{ij} but below the limit of quantification $y_{ij}^{cens} \in [LOD_{ij}, LOQ_{ij})$. The intervals could be $(-\infty, LOQ_{ij})$ (left censored data, as above) and $[LOQ_{ij}, +\infty)$ (right censored data).

—————————————————— MONOLIX ——————————————————

```
How MONOLIX handles BLQ data is described in Section 4.5.
```

A.6 Modeling the inter-occasion variability

We will denote y_{ikj} the jth observation for subject i during occasion k:

$$y_{ikj} = f(\psi_{ik}, t_{ikj}) + g(\psi_{ik}, t_{ikj}, \xi)\varepsilon_{ikj} \qquad (A.7)$$

Here, $\psi_{ik} = h(\varphi_{ik})$ is the individual parameter of subject i at occasion k:

$$\varphi_{ik} = C_{ik}\,\mu + \eta_i + \kappa_{ik} \qquad (A.8)$$

- C_{ik} is the matrix of covariates of subject i at occasion k,

- η_i random effect of subject i (inter-subject variability): $\eta_i \sim \mathcal{N}(0, \Omega)$,

- κ_{ik} random effect of subject i at occasion k (inter-occasion variability): $\kappa_{ik} \sim \mathcal{N}(0, \Gamma)$,

- η_i and κ_{ik} are assumed to be independent,

- Ω inter-subject variability covariance matrix,

- Γ inter-occasion variability covariance matrix.

A.7 Discrete data models

The basic model proposed in (A.1) is a regression model used for fitting continuous data that can be extended for categorical data or count data models. Assume that (y_{ij}) takes its values in $\{0, 1, 2, \ldots\}$. We define the conditional likelihood of the observations using a mixed effects model:

$$\mathbb{P}(y_{ij} = k | \psi_i) = f(k, x_{ij}, \psi_i) \ , \ 1 \le i \le N \ , \ 1 \le j \le n_i \qquad (A.9)$$

In other words, for any i, the probability that y_{ij} takes the value k depends on some (unknown) individual parameter ψ_i and possibly on some (known) design variable x_{ij}.

A mixed hidden Markov models (mixed HMM, or MHMM) assumes that there exists some non observed sequences (z_{ij}) (the states) that take their values in $1, 2, \ldots L$ such that, for any i,

- $(z_{ij}, j \ge 1)$ is a Markov Chain,

- conditionally to the sequence of states (z_{ij}), the (y_{ij}) are independent random variables

- the transition probabilities $\mathbb{P}(z_{i,j+1} = v | z_{ij} = u)$ and the emission probabilities (*i.e.* conditional probabilities) $\mathbb{P}(y_{ij} = k | z_{ij} = u)$ depend on some individual parameters ψ_i.

How to model categorical data, count data and HMM is described respectively in
Sections Section 4.7.1, Section 4.7.2 and Section 4.7.4.

A.8 Mixture models and model mixtures

A.8.1 Mixture models

In MONOLIX, a mixture model assume that there exist some "latent" categorical covariate G that takes K values. Then, the mixture model reduces to the categorical covariate model described Section A.2 but here, the categorical covariates are unknown: they are treated as random variables and the probabilities

$$\pi_k = \mathbb{P}(G_i = k)$$

are part of the statistical model and should be estimated as well.

A.8.2 Model mixtures

Let $f_1, f_2, \ldots f_K$ be K different structural models,

- **Between Subject Model Mixture (BSMM)**

  We assume that some categorical covariate G takes K values and that

  $$y_{ij} = f_k(x_{ij}, \psi_i) + \varepsilon_{ij} \quad , \quad \text{if} \quad G_i = k$$

  In a BSMM model, the "latent" categorical covariates are unknown: they are treated as random variables and the probabilities

  $$\pi_k = \mathbb{P}(G_i = k)$$

  are part of the statistical model and should be estimated as well.

- **Within Subject Model Mixture (WSMM)**

  For any patient i, let $p_{i,1}, p_{i,2}, \ldots, p_{i,K}$ be K proportions such that

  $$
  \begin{aligned}
  y_{ij} &= f_i(x_{ij}, \psi_i) + \varepsilon_{ij} \\
  f_i &= p_{i,1} f_1 + p_{i,2} f_2 + \ldots + p_{i,K} f_K
  \end{aligned}
  $$

  In a WSMM model, the proportions $(p_{i,k})$ are additional individual parameters that should be modeled as well (under the constraint that the sum is 1).

How to use mixture models is described in Section Section 4.10.1.
Examples of BSMM and WSMM with Monolix are presented in Section 4.10.2.

A.9 Prior models on fixed effects parameters

It is possible to define prior distribution models on the fixed effects. The allowed distributions:

- log-normal

- logit-normal

- probit-normal

- user-defined: transformation of a gaussian distribution:

$$h^{-1}(\mu) \sim \mathcal{N}(\mu_0, \sigma_\mu^2)$$

where h is any increasing function defined for all real numbers.

How to use priors on fixed effects is described in Section Section 4.3.

Appendix B

Preferences

This appendix provides the details of MONOLIX preferences. The preferences are defined in the preferences.xmlx file. The file is loaded when MONOLIX is started.

B.1 General

The user can modify the data to customize his own preferences.

1. Go to directory `monolixData` in users folder, then `/config/` and open the preference.xmlx file.

2. Modify the chosen fields, then save and restart MONOLIX if the software is opened.

If the user wants to find the default configuration again, the `preference.xmlx` file of `monolixData` folder must be deleted.

This chapter describes the correspondences between the file data and preferences in MONOLIX.

B.2 Graphic settings

Generally, data are visual settings of graphics.

There are 3 kinds of patterns:

1. matrix - *value* corresponds to the number of rows and *vector* contains all the elements of the matrix.

2. list of chars - *<charList>* tag contains a set of tags *<char>* which have chars as value.

3. numeric value - can be a string, an integer or a real.

B.2.1   Categorized Data

Data	Description	Type
fill_cat_color	Color of theoretical distribution line	matrix
fill_out_color	Outliers color	matrix
fill_CI	Color of confidence intervals	matrix
line_LineStyle	Lines style	value
line_LineWidth	Lines width	value
line_Color	Lines color	matrix
bins_Color	Color of bins limit lines	matrix
figure_background	Color of figure background	matrix

B.2.2   Covariates

Data	Description	Type
data_Color	Data colors	matrix
data_Marker	Data markers	List of characters
data_MarkerSize	Size of data markers	value
spline_LineStyle	Line style of spline	value
spline_LineWidth	Line width of spline	value
spline_Color	Line color of spline	matrix
regression_LineStyle	Style of regression line	value
regression_LineWidth	Width of regression line	value
regression_Color	Color of regression line	matrix
line_LineStyle	Lines style	value
line_LineWidth	Lines width	value
line_Color	Lines color	matrix
figure_background	Color of figure background	matrix

B.2.3 Parameters distribution

Data	Description	Type
histo_Color	Histogram color	matrix
paramEstimatedPDF_LineStyle	Style of non parametric pdf line	value
paramEstimatedPDF_LineWidth	Width of non parametric pdf line	value
paramEstimatedPDF_Color	Color of non parametric pdf line	matrix
predictedPdf_LineStyle	Style of population distribution line	value
predictedPdf_LineWidth	Width of population distribution line	value
predictedPdf_Color	Color of population distribution line	matrix
predictedEstimatedPDF_LineStyle	Style of theoretical median line	value
predictedEstimatedPDF_LineWidth	Width of theoretical median line	value
predictedEstimatedPDF_Color	Color of theoretical median line	matrix
confidence_LineStyle	Style of confidence intervals line	value
confidence_LineWidth	Width of confidence intervals line	value
confidence_Color	Color of confidence intervals line	matrix
figure_background	Color of figure background	matrix

B.2.4 Individual fits

Data	Description	Type
data_Color	Data colors	matrix
data_Marker	Data markers	List of characters
data_MarkerSize	Size of data markers	value
cens_Color	Censored data colors	matrix
cens_Marker	Censored data markers	List of characters
cens_MarkerSize	Size of censored data markers	value
populationFits_Color	Color of population fits line	matrix
populationFits_LineStyle	Style of population fits line	value
populationFits_LineWidth	Width of population fits line	value
IndividualFits_Color	Color of individual fits line	matrix
IndividualFits_LineStyle	Style of individual fits line	value
IndividualFits_LineWidth	Width of individual fits line	value
percentile	Percentile marker	value
confidence_pop_color	Color of prediction interval line	matrix
confidence_pop_linestyle	Style of prediction interval line	value
median_pop_linestyle	Style of median line	value
median_pop_color	Color of median line	matrix
figure_background	Color of figure background	matrix

B.2.5 Joint distribution

Data	Description	Type
data_Color	Data colors	matrix
data_Marker	Data markers	List of characters
data_MarkerSize	Size of data markers	value
spline_LineStyle	Line style of spline	value
spline_LineWidth	Line width of spline	value
spline_Color	Line color of spline	matrix
regression_LineStyle	Style of regression line	value
regression_LineWidth	Width of regression line	value
regression_Color	Color of regression line	matrix
figure_background	Color of figure background	matrix

B.2.6 Predictions vs observations

Data	Description	Type
data_Color	Data colors	matrix
data_Marker	Data markers	List of characters
data_MarkerSize	Size of data markers	value
cens_Color	Censored data colors	matrix
cens_Marker	Censored data markers	List of characters
cens_MarkerSize	Size of censored data markers	value
spline_LineStyle	Line style of spline	value
spline_LineWidth	Line width of spline	value
spline_Color	Line color of spline	matrix
regression_LineStyle	Style of regression line	value
regression_LineWidth	Width of regression line	value
regression_Color	Color of regression line	matrix
figure_background	Color of figure background	matrix
segments_LineStyle	Line style of segments	value
segments_LineWidth	Line width of segments	value
segments_Color	Line color of segments	matrix

B.2.7 Residuals

Data	Description	Type
data_Color	Data colors	matrix
data_Marker	Data markers	List of characters
data_MarkerSize	Size of data markers	value
cens_Color	Censored data colors	matrix
cens_Marker	Censored data markers	List of characters
cens_MarkerSize	Size of censored data markers	value
spline_LineStyle	Line style of spline	value
spline_LineWidth	Line width of spline	value
spline_Color	Line color of spline	matrix
line_LineStyle	Lines style	value
line_LineWidth	Lines width	value
line_Color	Lines color	matrix
bins_Color	Color of bins limit lines	matrix
plot_emp_color	Color of empirical percentile line	matrix
plot_emp_linestyle	Style of empirical percentile line	List of characters
plot_emp_linewidth	Width of empirical percentile line	value
plot_emp_marker	Marker of outlier dots	value
plot_emp_markersize	Size of outlier dots marker	value
plot_out_color	Color of outlier dots	matrix
plot_sim_color	Color of theoretical percentile line	matrix
plot_sim_linewidth	Width of theoretical percentile lines	value
plot_sim_linestyle	Style of theoretical percentile lines	List of characters
fill_cat_color	Color of confidence interval	matrix
fill_out_color	Color of outliers area	matrix
regression_LineStyle	Style of regression line	value
regression_LineWidth	Width of regression line	value
regression_Color	Color of regression line	matrix
histo_Color	Histogram color	matrix
empdens_LineStyle	Style of empirical pdf line	value
empdens_LineWidth	Width of empirical pdf line	value
empdens_Color	Color of empirical pdf line	matrix
thdens_LineStyle	Style of theoretical empirical pdf line	value
thdens_LineWidth	Width of theoretical empirical pdf line	value
thdens_Color	Color of theoretical empirical pdf line	matrix
figure_background	Color of figure background	matrix

B.2.8 Spaghetti

Data	Description	Type
data_Color	Data colors	matrix
data_Marker	Data markers	List of characters
data_MarkerSize	Size of data markers	value
cens_Color	Censored data colors	matrix
cens_Marker	Censored data markers	List of characters
cens_MarkerSize	Size of censored data markers	value
line_LineStyle	Lines style	value
line_LineWidth	Lines width	value
figure_background	Color of figure background	matrix

B.2.9 Prediction distribution

Data	Description	Type
data_Color	Data colors	matrix
data_Marker	Data markers	List of characters
data_MarkerSize	Size of data markers	value
fill_shading	Prediction distribution area	matrix
fill_shading_discrete	Prediction distribution area in discrete case	matrix
line_LineStyle	Lines style	value
line_LineWidth	Lines width	value
bins_Color	Color of bins limit lines	matrix
plot_emp_color	Color of empirical percentile line	matrix
plot_emp_linestyle	Style of empirical percentile line	List of characters
plot_emp_linewidth	Width of empirical percentile line	value
median_color	Color of median line	matrix
figure_background	Color of figure background	matrix

For `fill_shading` and `fill_shading_discrete` settings, a 2×3 matrix is necessary, that corresponds to the two limit colors used to fill the areas.

B.2.10 VPC

Data	Description	Type
data_Color	Data colors	matrix
data_Marker	Data markers	List of characters
data_MarkerSize	Size of data markers	value
cens_Color	Censored data colors	matrix
cens_Marker	Censored data markers	List of characters
cens_MarkerSize	Size of censored data markers	value
line_LineStyle	Lines style	value
line_LineWidth	Lines width	value
line_Color	Lines color	matrix
plot_emp_color	Color of empirical percentile line	matrix
plot_emp_linestyle	Style of empirical percentile line	List of characters
plot_emp_linewidth	Width of empirical percentile line	value
plot_emp_marker	Marker of outlier dots	value
plot_emp_markersize	Size of outlier dots marker	value
plot_out_color	Color of outlier dots	matrix
plot_sim_color	Color of theoretical percentile line	matrix
plot_sim_marker	Marker of theoretical percentile	matrix
plot_sim_markersize	Size of theoretical percentile marker	matrix
plot_sim_linewidth	Width of theoretical percentile lines	value
plot_sim_linestyle	Style of theoretical percentile lines	List of characters
fill_cat_color	Color of confidence interval	matrix
fill_out_color	Color of outliers area	matrix
figure_background	Color of figure background	matrix

B.2.11 NPC - BLQ

Data	Description	Type
fill_cat_color	Color of theoretical CDF	matrix
fill_out_color	Color of median line	matrix
fill_CI	Color of confidence intervals	matrix
line_LineStyle	Lines style	value
line_LineWidth	Lines width	value
line_Color	Lines color	matrix
figure_background	Color of figure background	matrix

B.2.12   Time to event (Kaplan-Meier)

Data	Description	Type
figure_background	Color of figure background	matrix
fill_shading	Confidence interval area	matrix
fill_shading_average	Confidence interval area for average	matrix
median_color	Color of median line	matrix
median_lineStyle	Line style of median	value
median_lineWidth	Line width of median	value
median_average_color	Color of average median	matrix
median_average_lineStyle	Line style of average median	value
median_average_lineWidth	Line width of average median	value
cens_Color	Censored data colors	matrix
cens_Marker	Censored data markers	List of characters
cens_MarkerSize	Size of censored data markers	value
average_color	Average line color	matrix
average_lineStyle	Line style of average	value
average_lineWidth	Line width of average	value
survival_color	Survival function line color	matrix
survival_lineStyle	Survival function line style	value
survival_lineWidth	Survival function line width	value

For `fill_shading` setting, a 2×3 matrix is necessary, that corresponds to the two limit colors used to fill the areas.

B.2.13   Transition probabilities

Data	Description	Type
figure_background	Color of figure background	matrix
line_LineStyle	Lines style	value
line_LineWidth	Lines width	value
bins_Color	Color of bins limit lines	matrix
line_LineStyle	Lines style	value
line_LineWidth	Lines width	value
line_Color	Lines color	matrix

B.2.14   Prior distribution

Data	Description	Type
figure_background	Color of figure background	matrix
confidence_LineStyle	Style of confidence intervals line	value
confidence_LineWidth	Width of confidence intervals line	value
confidence_Color	Color of confidence intervals line	matrix
histo_Color	Histogram color	matrix
nonParamPDF_LineStyle	Style of non parametric pdf line	value
nonParamPDF_LineWidth	Width of non parametric pdf line	value
nonParamPDF_Color	Color of non parametric pdf line	matrix
priorPDF_LineStyle	Style of prior distribution line	value
priorPDF_LineWidth	Width of prior distribution line	value
priorPDF_Color	Color of prior distribution line	matrix
medianPDF_LineStyle	Style of theoretical median line	value
medianPDF_LineWidth	Width of theoretical median line	value
medianPDF_Color	Color of theoretical median line	matrix

B.2.15   Individual contribution

Data	Description	Type
figure_background	Color of figure background	matrix
bar_Color	Color of bars	matrix

For `bar_Color` setting, a 2×3 matrix is necessary, that corresponds to the two likelihoods.

B.2.16   Convergence of SAEM

Data	Description	Type
figure_background	Color of figure background	matrix

B.3   Session related settings

B.3.1   session

Field	Description	Type
timestamp_value	Use timestamping	value
historic_size	Size of historic file (projects, models)	value
nbDatasetRows	(see nbShownColumns in dataSelection gui)	value
save_graphics_format	Printed graphics format	value
lockModels	Lock possibility to modify models	value
modelFilter	Filter of structural model	'none', 'm' or 'mlxtran'
useCurrentFolderByDefault	Use current folder by default	value
editor	Editor to use	editor path

B.3.2   project

Field	Description	Type
dosesToAddForSteadyState	number of doses to simulate Steady state	value

B.3.3   gui

datasetSelection

Field	Description	Type
nbShownColumns	number of columns to show	value
nbRows	number of rows to show (maximum 10)	value

Bibliography

[1] ALLASSONNIÈRE, S., KUHN, E., AND TROUVÉ, A. Construction of Bayesian deformable models via stochastic approximation algorithm: A convergence study. *Bernoulli (to appear)* (2010).

[2] BERTRAND, J., COMETS, E., AND MENTRÉ, F. Detecting a gene effect in pharmacokinetic models: comparison of different methods (poster). *PAGE, Brugge* (2006).

[3] CHAN, P. AND JACQMIN, P., LAVIELLE, M., MCFADYEN, L., AND WEATHERLEY, B. The use of the SAEM algorithm in MONOLIX software for estimation of population pharmacokinetic-pharmacodynamic-viral dynamics parameters of maraviroc in asymptomatic HIV subjects. *Journal of Pharmacokinetics and Pharmacodynamics (to appear)* (2010).

[4] COMETS, E., VERSTUYFT, C., LAVIELLE, M., JAILLON, P., BECQUEMONT, L., AND MENTRE, F. Modelling the influence of MDR1 polymorphism on digoxin pharmacokinetic parameters. *European Journal of Clinical Pharmacology 63* (2007), 437–449.

[5] DAVIDIAN, M., AND GILTINAN, D. *Nonlinear Models for Repeated Measurement Data.* Chapman and Hall, 1995.

[6] DAVIDIAN, M., AND GILTINAN, D. Nonlinear models for repeated measurements: An overview and update. *JABES 8* (2003), 387–419.

[7] DELATTRE, M., DEL MORAL, P., AND LAVIELLE, M. The SAEM algorithm in MONOLIX for non-linear mixed effects models with stochastic differential equations (poster). *PAGE, Berlin* (2010).

[8] DELATTRE, M., SAVIC, R., MILLER, R., KARLSSON, M., AND LAVIELLE, M. Estimation of mixed hidden Markov models with SAEM. Application to daily seizures data. (oral presentation). *PAGE, Berlin* (2010).

[9] DELYON, B., LAVIELLE, M., AND MOULINES, E. Convergence of a stochastic approximation version of the EM algorithm. *Ann. Statist. 27*, 1 (1999), 94–128.

[10] DONNET, S., AND SAMSON, A. Estimation of parameters in incomplete data models defined by dynamical systems. *Journ. of Stat. and Plan. Infer. 50* (2007), 2381–2398.

[11] GIRARD, P., AND MENTRÉ, F. A comparison of estimation methods in nonlinear mixed effects models using a blind analysis (oral presentation). *PAGE, Pamplona* (2005).

[12] JAFFRÉZIC, F., MEZA, C., FOULLEY, J., AND LAVIELLE, M. The SAEM algorithm for the analysis of nonlinear traits in genetic studies. *Genetics Selection Evolution 38* (2006), 583–600.

[13] KUHN, E., AND LAVIELLE, M. Coupling a stochastic approximation version of EM with a MCMC procedure. *ESAIM P&S 8* (2004), 115–131.

[14] KUHN, E., AND LAVIELLE, M. Maximum likelihood estimation in nonlinear mixed effects models. *Computational Statistics and Data Analysis 49*, 4 (2005), 1020–1038.

[15] LAVIELLE, M., AND KUHN, E. Maximum likelihood estimation in nonlinear mixed effects models (oral communication). *PAGE, Verona* (2003).

[16] LAVIELLE, M., AND MENTRÉ, F. Estimation of population pharmacokinetic parameters of saquinavir in HIV patients and covariate analysis with MONOLIX (poster). *PAGE, Pamplona* (2005).

[17] LAVIELLE, M., AND MENTRÉ, F. Estimation of population pharmacokinetic parameters of saquinavir in HIV patients with the MONOLIX software. *Journal of Pharmacokinetics and Pharmacodynamics 34*, 2 (2007), 229–249.

[18] LAVIELLE, M., MESA, H., CHATEL, K., AND VERMEULEN, A. Mixture models and model mixtures with MONOLIX (oral presentation). *PAGE, Berlin* (2010).

[19] MAKOWSKI, D., AND LAVIELLE, M. Using SAEM to estimate parameters of models of response to applied fertilizer. *Jour. of Agr., Bio, and Env. Stat. 11*, 1 (2006), 45–60.

[20] PANHARD, X., AND SAMSON, A. Extension of the SAEM algorithm for the estimation of inter-occasion variability: application to the population pharmacokinetics of nelfinavir and its metabolite m8 (poster). *PAGE, Brugge* (2006).

[21] PINHEIRO, J. C., AND BATES, D. M. *Mixed-Effects Models in S and S-PLUS*. Springer, New York, 2000.

[22] SAMSON, A., LAVIELLE, M., AND MENTRÉ, F. Approximation EM algorithm in nonlinear mixed effects models: an evaluation by simulation (oral communication). *PAGE, Uppsala* (2004).

[23] SAMSON, A., LAVIELLE, M., AND MENTRÉ, F. Extension of the SAEM algorithm to left-censored data in nonlinear mixed-effects model: application to HIV dynamics model. *Computational Statistics and Data Analysis 51* (2006), 1562–1574.

[24] Samson, A., Lavielle, M., and Mentré, F. The SAEM algorithm for non-linear mixed models with left-censored data and differential systems: application to the joint modeling of hiv viral load and cd4 dynamics under treatment (oral presentation). *PAGE, Brugge* (2006).

[25] Samson, A., Lavielle, M., and Mentré, F. The SAEM algorithm for group comparison tests in longitudinal data analysis based on nonlinear mixed-effects model. *Stat. in Med. 26* (2007), 4860–4875.

[26] Samson, A., Panhard, X., Lavielle, M., and Mentré, F. Generalisation of the SAEM algorithm to nonlinear mixed effects model defined by differential equations: application to HIV viral dynamic models (poster). *PAGE, Pamplona* (2005).

[27] Savic, R., and Lavielle, M. A new SAEM algorithm: Performance in population models for count data. *Journal of Pharmacokinetics and Pharmacodynamics 36* (2009), 367–379.

[28] Savic, R., Mentre, F., and Lavielle, M. Implementation and evaluation of an SAEM algorithm for longitudinal ordered categorical data with an illustration in pharmacometrics. *The AAPS Journal (to appear)* (2010).

[29] Snoeck, E., Chanu, P., Lavielle, M., Jacqmin, P., Jonsson, N., Jorga, K., Goggin, T., Jumbe, S., and Frey, N. Hepatitis C viral dynamics explaining breakthrough, relapse or response after chronic treatment. *Clinical Pharmacology and Therapeutics (AAPS Outstanding Manuscript Award in Modeling and Simulation) 87* (2010), 706–713.