Pirana

The pharmacometrician's workbench

Version $\geq$ 2.10.0
Installation guide and Manual

March 29, 2015

Contents

1 Introduction **3**

2 Pirana installation **4**

2.1 Required / recommended software 4

2.2 Installation 5

2.2.1 Installation procedure on Windows 5

2.2.2 Installation procedure on Mac OS X 5

2.2.3 Installation procedure on Linux 5

2.3 Installation of license file 7

2.4 Configuration 7

2.5 Configuration for modeling groups 9

3 Using Pirana **11**

3.1 Overview 11

3.1.1 Basic functions 11

3.1.2 Model management 12

3.1.3 Projects 14

3.1.4 Data files 14

3.2 Working with NONMEM 15

3.2.1 Managing NONMEM installations 15

3.2.2 Setting environment variables 16

3.2.3 Running models 17

3.3 Analyzing results & output 21

3.3.1 Main overview 21

3.3.2 Parameter estimates 21

3.3.3 Run Reports 22

3.3.4 Intermediate results 23

3.3.5 Run Records 24

3.3.6 Data Inspector 25

3.3.7 Visual run records 26
3.3.8 R scripting for creating graphs and file processing 26
3.3.9 Xpose support 28
3.3.10 Wizards 29
3.4 Model translation 30
3.5 Batch operations 31
3.6 Miscellaneous functionality 32
3.7 Conventions and Methods 36
3.8 Keyboard shortcuts 37
3.9 Command line parameters 38
3.10 Troubleshooting / FAQ 39
3.11 Monolix 41

4 Pirana and clusters **42**
4.1 Clusters 42
4.1.1 Method 1: Server-based installation 42
4.1.2 Method 2: Local installation 43
4.2 Interaction with job schedulers 44
4.2.1 Working with the SGE 45
4.3 PCluster 45

5 Automated modeling workflow in Pirana **48**
5.1 Background 48
5.2 Tutorial 48

6 cPirana **55**
6.1 Installing cPirana 55
6.1.1 Using cPirana 55

7 Validation **57**
7.1 Pirana validation tool 58
7.2 PsN validation tool 59

8 Endnotes **60**
8.1 Acknowledgements 60
8.2 Bug reporting / suggestions 60

Chapter 1

Introduction

Pirana is a modeling workbench for NONMEM, PsN and Xpose, offering a graphical user interface and many auxiliary tools to support modeling & simulation analyses.

Development of Pirana was started in 2007, and development still continues. Pirana is currently released under a commercial license for non-academic users, while it is released (free of charge) under a Creative Commons license for academic users. For the detailed license, please see the license document supplied with each Pirana version.

Pirana is designed to be very flexible, and extendible: it integrates with many existing software such as R, Excel, and Berkeley Madonna, and it runs on all major operating systems. We aim for Pirana to be very intuitive, but browsing through this manual before you start working with Pirana is recommended. For many common functionality in Pirana, a Quick Guide is available (from Pirana's help-menu, or from www.pirana-software.com). Also have a look at our website for more information and an FAQ.

Please note that since 2014, a web-version of Pirana is also available: *PiranaJS*. The aim of this web-app is to provide a fast and easily accessible modeling workbench for use on clusters and in the cloud. PiranaJS is released under the same commercial and academic licenses, more information is available on our website.

Please do not hesitate to contact us if you have any questions about Pirana or any of the other tools developed by us.

info@pirana-software.com

Chapter 2

Pirana installation

Software requirements are summarized below, followed by some additional details on the installation procedure.

2.1 Required / recommended software

Although the only real requirement is an installation of NONMEM, some additional software is highly recommended for optimal use of Pirana, while other softwares might be useful as well.

NONMEM Pirana can use both standard 'from-CD'-installation of NONMEM and NMQual NONMEM installations. NONMEM doesn't have to be installed on your local PC, since Pirana can also connect to other PCs or clusters.

R This open-source software can be obtained from http://www.r-project.org/, and is highly recommended for optimal use of Pirana. R Studio (http://www.r-studio.org/) is a powerful GUI for R that works well with Pirana. Severall R libraries need to be installed to be able to use all the R scripts that are included in the Scripts library in Pirana, such as: lattice, ggplot2, MASS.

PsN Strictly, Pirana does not require PsN installed, although the PsN-toolkit is highly recommended. The latest version of PsN can be obtained from http://psn.sourceforge.net/.

Xpose This R-package for model diagnostics is recommended and can be obtained from http://xpose.sourceforge.net. An Xpose GUI is available within Pirana.

WFN Not required/recommended. Pirana offers only basic support for Wings for NONMEM.

NMQual Recommended for keeping an audit trail of NM installations and autamatically performing bug-fixes. Pirana supports the use of NMQual NONMEM installations. NMQual also requires Perl and the module XML::XPath installed. More details can be found at the Metrum website

2.2 Installation

2.2.1 Installation procedure on Windows

Download the installer from the Pirana website, and install to any location on your hard-drive. Before exploring Pirana's functions, you should first check your settings (*File* → *Settings* → *General...*) and software integration (*File* → *Settings* → *Software locations...*). If you want to connect to clusters, also update the *Cluster* settings. Pirana is tested on XP, Vista, 7, and 8. Upon installation, Windows or a virus scanner may complain that installation is not safe, but this warning can be ignored, we make every effort to provide virus-free software.

2.2.2 Installation procedure on Mac OS X

On Mac, you should first install the Xcode tools, which are included as optional installs on the (Snow) Leopard install DVDs. In the most recent OSX versions, Xcode can be installed from the App Store. After installing Xcode, make sure you also install the Command Line Tools (From within Xcode, go to *Preferences*, and *Downloads*). Secondly you will have to install an X-window manager. For old versions of OSX, X11 is most likely already installed. If you are running Lion or later, you will however have to install XQuartz instead of X11, which is downloadable online (google for 'XQuartz').

Pirana for Mac is distributed as an executable, so no Perl installation is required, but it is also possible to run from source (see Linux explanation). When opening Pirana for the first time, your system may complain that Pirana is not safe, since it is not installed from the App Store. In this case, you will have to set your security settings in your systems *Preferences* to run apps from *Everywhere*, instead of from *App Store only*.

2.2.3 Installation procedure on Linux

For Linux, an executable is made available as well. This executable is compiled on a 32-bit system. If for some reason this executable doesn't run on your system, the Perl source-code can be executed directy in Perl. This requires manual installation of a few additional libraries and modules, see below.

Installing Perl and X11 development libraries

For Pirana to be able to create the GUI, the X11 development libraries (`libX11-dev`) should be installed, as well as the Perl/Tk module. In Ubuntu / Debian, you can use the Synaptic package manager to install these, or using apt from the shell:

```
sudo apt-get install libX11-dev perl-tk
```

Installing additional Perl modules

Pirana makes use of a number of publicly available Perl-modules, which should also be installed. Some of these modules are likely to be already installed with you current Perl distribution, while others have to be installed manually. Below is a short guidance on how to install these modules. Further guidance on installing Perl modules can be found here: http://www.cpan.org/modules/INSTALL.html). To make a connection to the Perl module archive (CPAN), type:

```
sudo perl -MCPAN -e shell
```

The following commands may be needed to set up the CPAN shell to be able to correctly `make` the modules into your Perl distribution.

```
o conf make /usr/bin/make
o conf make_install_make_command 'sudo make'
o conf commit
```

Next, use the `install` command to install required modules into your Perl distribution (mind the case-sensitivity for the module names). Look in Some of these may already be installed, which will be reported as such. If you cannot install some modules from CPAN directly, you have to download and install these modules (and their dependencies) manually. Look in the file pirana.pl to see which modules to install (this may change between Pirana versions).

```
install Tk::PlotDataset
install Tk::JComboBox
install ...
```

Some required modules cannot be installed directly from CPAN. These modules are supplied with Pirana (in the folder `/packages`) and should be installed manually. From within each of the two package folders, execute in a shell:

```
perl Makefile.PL
make
sudo make install
```

Note: Checking whether Perl modules are installed correctly can be done by executing the following in the terminal window, e.g.: `perl -e 'use Tk'` which should result in no error messages.

Pirana installation

After installing these Perl modules, copy the entire pirana folder contained in the zip-file to e.g. your home folder (`/home/username/`) or `/opt/pirana/` if you are system admin. Make sure that all perl files in that folder have execution rights. To grant these rights to yourself you can execute the following in the shell from within the Pirana folder:

```
sudo chmod 711 -R *
```

Pirana execution

Pirana can now be started from the command line using

```
perl pirana.pl
```

from within the Pirana folder. Pirana was tested on Ubuntu (9.04-12.04), OpenSUSE (11.1), and Arch Linux, with various Perl distributions. Pirana should work on any Linux distribution with X-windows and Perl/Tk installed.

2.3 Installation of license file

License files can be installed by going to *Help* → *Import license file*. On Windows, you can also install the license file by dropping it on the Pirana main window. Upon starting Pirana, the presence of a valid license file (`pirana.lic`) in Pirana's main installation folder will be checked. If no valid license file is present, a message will be displayed and some functionality of Pirana will be disabled. Academic, commercial, or trial license files can be obtained on the website.

2.4 Configuration

Although most preferences will be correct by default, we recommend to check at least the settings detailed below. Familiarize yourself with the other options as well, to get the most out of Pirana. Especially check the correct file extensions for NONMEM model files and output files.

File extension of models NONMEM control streams/model files. Default is `.mod`. Note that multiple model file extensions can be specified separated by a comma, e.g. `mod,ctl`.

File extension of results NONMEM output. Default is `.lst`

Software settings Pirana needs to know where other important software is installed, which is specified under *Software* from the *File* menu. References to software that you do not have installed, may be disregarded as they are ignored by Pirana.

Code editor Preferably an editor syntax-highlighting. We recommend the use of Emacs (all OS), Sublime Text 2/3 (all OS), PSPad (Windows), ConTEXT (Windows). If none is entered, or a non-existing program is specified, Pirana will use its built-in NM-TRAN editor.

R location The location (folder) of R, e.g. `C:/Program Files/R/R-3.1.0`. On Windows, at first start-up of Pirana it will search for the latest version of R that is installed, and automatically updates this setting accordingly. If R is installed in a non-standard location, please update this.

R GUI The GUI to be used for R-scripts. Recommended for this is RStudio or Emacs/ESS, but the RGUI supplied with the R distribution can also be used.

spreadsheet The location of your spreadsheet application, e.g. Excel or Gnumeric. Pirana tries to find your spreadsheet automatically.

Note: on Mac OSX you can either specify the application name (e.g. "Microsoft Excel"), or the actual location of the application (e.g. `/usr/local/bin/emacsclient`).

2.5 Configuration for modeling groups

IT sysadmins that want to distribute Pirana to a modeling group with pre-specified settings can do so by editing the files in the folder `ini_defaults` before distributing Pirana. This will allow users to start with appropriate defaults for Pirana's setting. Also, one or more clusters can be added by default, so that users do not have to add these themselves (and only have to update their username and login). Look in the file `/ini_defaults/clusters/readme.txt` for further information.

Configuration files location

If even more control is required, i.e. if the end-user should not be allowed to change part of the configuration at all, it is possible to change the location of the configuration files from the user's home directory to a different, protected shared location. The default location for Pirana's configuration files can be overriden by using the file `ini_locations.ini` located in the folder where you installed Pirana. Change the central ini-files rights to read-only, to be sure users do not change the central settings. More information about how to use this functionality is available in the annotated `ini_locations.ini` file itself.

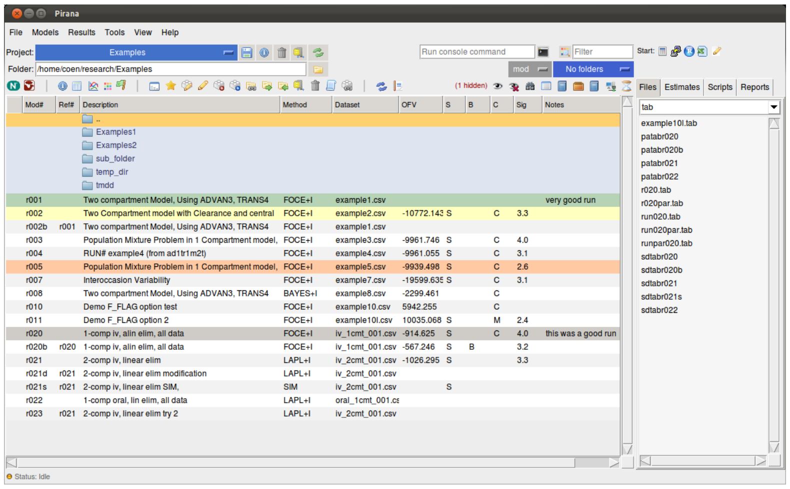
Chapter 3

Using Pirana

3.1 Overview

3.1.1 Basic functions

The Pirana window consists of a large area showing an overview of models in the current folder, and a smaller area on the right. The main overview acts as an *electronic lab-notebook* for modeling analyses. The list on the right shows e.g. datasets, R-scripts, or all files in the active folder, or alternatively a list of parameter estimates or reports, or a list of Git/SVN commits. Most buttons in Pirana's main screen are accompanied by a short description which is displayed if hovered above with the mouse pointer. By selecting a model or (data-) file in either lists and right-clicking the mouse, a menu with actions on models or run results is shown. Most of Pirana's functionality is available from this menu, and most options are also available from the toolbar (*View* $\rightarrow$ *Show toolbar*).


Pirana
File Models Results Tools View Help
Run console command
Filter
Start:
(1 hidden)
Files Estimates Scripts Reports

Project: Examples

Folder: /home/coen/research/Examples

mod | No folders

Mod#	Ref#	Description	Method	Dataset	OFV	S	B	C	Sig	Notes
		..								
		Examples1								
		Examples2								
		sub_folder								
		temp_dir								
		tmdd								
r001		Two compartment Model, Using ADVAN3, TRANS4	FOCE+I	example1.csv						very good run
r002		Two Compartment model with Clearance and central	FOCE+I	example2.csv	-10772.143	S		C	3.3	
r002b	r001	Two compartment Model, Using ADVAN3, TRANS4	FOCE+I	example1.csv						
r003		Population Mixture Problem in 1 Compartment model,	FOCE+I	example3.csv	-9961.746	S		C	4.0	
r004		RUN# example4 (from ad1tr1m2t)	FOCE+I	example4.csv	-9961.055	S		C	3.1	
r005		Population Mixture Problem in 1 Compartment model,	FOCE+I	example5.csv	-9939.498	S		C	2.6	
r007		Interoccasion Variability	FOCE+I	example7.csv	-19599.635	S		C	3.1	
r008		Two compartment Model, Using ADVAN3, TRANS4	BAYES+I	example8.csv	-2299.461			C		
r010		Demo F_FLAG option test	FOCE+I	example10.csv	5942.255			C		
r011		Demo F_FLAG option 2	FOCE+I	example10l.csv	10035.068	S		M	2.4	
r020		1-comp iv, alin elim, all data	FOCE+I	iv_1cmt_001.csv	-914.625	S		C	4.0	this was a good run
r020b	r020	1-comp iv, alin elim, all data	FOCE+I	iv_1cmt_001.csv	-567.246	S	B		3.2	
r021		2-comp iv, linear elim	LAPL+I	iv_2cmt_001.csv	-1026.295	S			3.3	
r021d	r021	2-comp iv, linear elim modification	LAPL+I	iv_2cmt_001.csv						
r021s	r021	2-comp iv, linear elim SIM,	SIM	iv_2cmt_001.csv		S				
r022		1-comp oral, lin elim, all data	LAPL+I	oral_1cmt_001.cs						
r023	r021	2-comp iv, linear elim try 2	LAPL+I	iv_2cmt_001.csv						

tab

- example10l.tab
- patabr020
- patabr020b
- patabr021
- patabr022
- r020.tab
- r020par.tab
- run020.tab
- run020par.tab
- runpar020.tab
- sdtabr020
- sdtabr020b
- sdtabr021
- sdtabr021s
- sdtabr022

Status: Idle

Figure 3.1: Pirana main screen (on Linux)

3.1.2 Model management

The main model overview is where all models and subdirectories in the current working directory are displayed. Only models are displayed that have a file-extension corresponding to the file extension specified in the *preferences*. If multiple file extensions were specified, choose the one you want to use in the current folder by selecting it from the listbox (the one next to the folder selector, on the right above the main models list). When models are double-clicked, the control stream is opened in the code-editor (if specified), or else in Pirana's built-in NM-TRAN editor.

Model views

By default, the model overview is shown as a list, listing all models ordered by run number. It is advised, but not mandatory, that models are named as a number (e.g. `001.mod`), or prepended with `run` (e.g. `run1.mod`, or `run001.mod`), see *Conventions and Methods* for more information. By default, the list is shown in *condensed mode*, meaning that for every model, a single row is used in the table. The list can however also be shown in *expanded mode*, which allows for longer model descriptions and notes in the overview. Additionally, in this mode all estimation methods are shown, while in condensed mode only the last estimation method and associated OFV is shown.

An alternate view mode is *tree view*, in which model development is shown as a hierarchical tree. The tree is built using parent/reference information included in the model files (see 'run record' explanation in this manual). When creating models in Pirana, this information is added automatically, and adheres to PsN's run record syntax.

The columns that are shown in the main overview can be activated or de-activated from the *View* → *Show columns* menu. Models can be filtered using the 'Filter' above the main overview table, or by colors or flags.

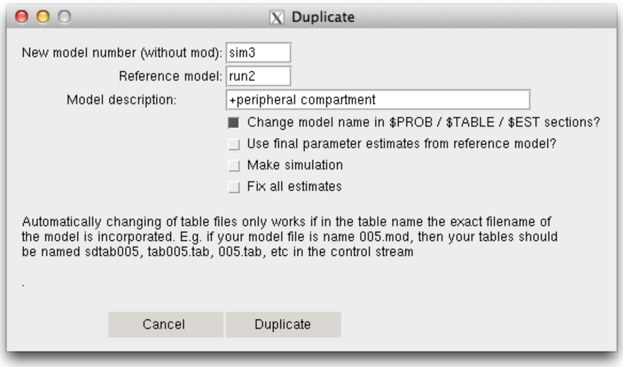
Model actions

New models can be created from scratch, from a template, by using the Wizards, or by duplicating an existing model:

Wizards Models can be created by using the PK model wizard. Choose the desired model type and estimation method, and a basic NONMEM model file will be created.

Templates Many basic template models are included. It is possible to build your own library with base models that you often use. Templates can be added by copying a model file to `/templates` in the Pirana directory. The template models should have the same file extension as your model files to be recognized as a template.

Duplication Duplication of models can be performed by selecting the parent model and clicking 'duplicate' from the context-menu (right click on the model). Optionally, final parameter estimates from the reference model can be updated in the new model, and also model-file numbers in `$TABLE` and `$EST` records. Some basic syntax rules should be adhered to ensure correct interpretation of final estimates, see Conventions and Methods at the end of this chapter. A model can also be *Duplicated for MSF restart*. This means that the model file is duplicated, but an $MSFI record is added, parameter estimate blocks are commented out, and the $MSFO record is updated.



New model number (without mod): sim3
Reference model: run2
Model description: +peripheral compartment
■ Change model name in $PROB / $TABLE / $EST sections?
☐ Use final parameter estimates from reference model?
☐ Make simulation
☐ Fix all estimates

Automatically changing of table files only works if in the table name the exact filename of the model is incorporated. E.g. if your model file is name 005.mod, then your tables should be named sdtab005, tab005.tab, 005.tab, etc in the control stream

.

Figure 3.2: Duplicate model

Automated modeling workflow From version 2.10.0, an *automated modeling workflow* is available in Pirana, in which models can be generated from a library. The aim of this functionality is to streamline the analysis of a new dataset, by fitting a range of models to it. This functionality is explained in more detail in a separate section.

Notes, flags and colors

To each model or run in Pirana, you can attach notes, flags, and colors. The colors can indicate e.g. *key runs*, *good runs*, or *bad runs*, but of course the meaning of the flags and color-coding is all up to the user. The notes and color info are stored in a database file (*pirana.dir*), which is created automatically in each folder that holds models. To add notes to a model, select the model, right click and select *Model* → *Notes and info*, or using the *Ctrl-I* shortcut. Models and results can be given a color by selecting the model, right-clicking, and selecting the desired color/flag from the *Colors & flags* submenu. Pirana also supports filtering of models/runs by color.

Note: In each active folder that is visited with Pirana, a small SQLite database is created ('pirana.dir') which is able to store information about models. So if you archive your projects manually, make sure to include these files as well.

3.1.3 Projects

Pirana allows you to save a link to a folder as a project, which will then be shown in the blue optionmenu (above the active folder entry). This allows you to quickly switch to the directory that is linked to that project. To add a project to the list, browse to a folder by clicking on the folder-icon next to the location bar, or by clicking through the directory-listing in the model overview. Next, click on the *disk*-icon next to the project name, give your project a unique name and press *Save*. Your project is now available from the listbox. To delete a project from the list, click the trash icon next to the project list. The green refresh-icon refreshes the view of the current directory, and should be applied when you make changes to models or add files outside of Pirana. Also when a run is finished, you should refresh to gather the results into Pirana.

3.1.4 Data files

The list on the right of the screen shows files in the current folder, if *Files* is selected as the active tab. Pirana can show tab-files, csv-files, R scripts, Xpose files, and other files, which can be selected from the list above. It is also possible to specify your own filter. Right-clicking on a selected file shows a menu with possible actions on the file.

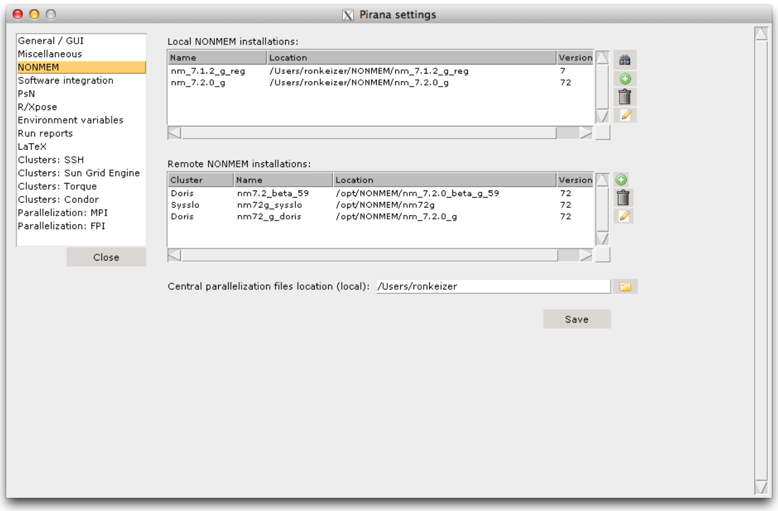
Note: When the Xpose option is chosen, only unique run numbers are shown, instead of all tabualar data files. After selecting an Xpose dataset, click the 'Open in R' the right-click-menu, and R read in the datasets and create the Xpose object.

3.2 Working with NONMEM

There are several ways in which NONMEM can be used from Pirana. The first one is to use the `nmfe`-script supplied with NONMEM. For this, you have to instruct Pirana where NONMEM is installed. The other (recommended) way to run NONMEM, is to use PsN. When you run NONMEM through PsN, you don't have to tell Pirana where NONMEM is located, since this is already specified in the `psn.conf` file of PsN.

3.2.1 Managing NONMEM installations

Existing NONMEM installations can be added to Pirana via the Settings menu, under the NONMEM tab. Here, both local (upper) and cluster (lower) installations can be added for use in Pirana. A *smart search* tool is implemented for local NONMEM installations, which searches for NONMEM installations in the most common locations on your local drives. If you have installed NONMEM in a non-common location, or want to use NONMEM on a remote cluster, add the paths to NONMEM manually.



Local NONMEM installations:

Name	Location	Version
nm_7.1.2_g_reg	/Users/ronkeizer/NONMEM/nm_7.1.2_g_reg	7
nm_7.2.0_g	/Users/ronkeizer/NONMEM/nm_7.2.0_g	72

Remote NONMEM installations:

Cluster	Name	Location	Version
Doris	nm7.2_beta_59	/opt/NONMEM/nm_7.2.0_beta_g_59	72
Sysslo	nm72g_sysslo	/opt/NONMEM/nm72g	72
Doris	nm72_g_doris	/opt/NONMEM/nm_7.2.0_g	72

Central parallelization files location (local): /Users/ronkeizer

Figure 3.3: NONMEM settings window

3.2.2 Setting environment variables

NONMEM requires a Fortran compiler installed. For this compiler to function properly, it is important to set the environment variables correctly. Especially for Intel compilers, this can be trooublesome, as different environmental variables need to be defined. For GNU compilers, setting the PATH environment variable is usually sufficient. There are several ways you have control over the environment variables when using Pirana:

1. In the settings menu, under *Environmental variables*, the PATH variable used within the Pirana environment can be defined, or additional folders can be added to the existing PATH. Also, additional environmental variables can be defined here.

2. Alerntatively, in the same settings screen for each nmfe-run that is started, you can specify a command that will be executed before starting nmfe-type runs, which can be e.g.

   ```
     PATH=%PATH%;C:\gfortran\bin
   ```

3. At startup, Pirana will check for the existence of 2 files in the Pirana base folder: *set_env.txt* and *add_env.txt*. These files can be used to either set, or add to the system variables, respectively. The files may look e.g. like this:

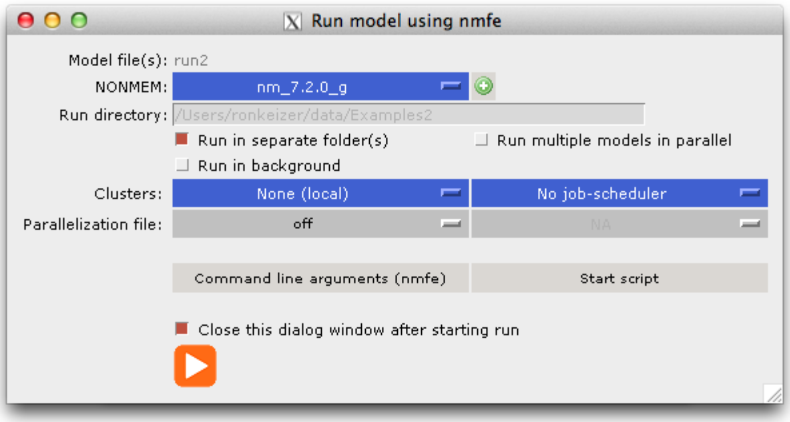
   ```
   PATH=C:\nmvi\run;C:\MinGW\bin
   VARX=C:\bladibla;etc
   ```

3.2.3 Running models

As mentioned before, a model can be run using `nmfe`, *PsN* or *WFN* (WFN is Windows only). This can be done by selecting a model from the list, and right-clicking to show the context-menu. From here, you can either select `nmfe`, or the PsN or WFN options. The commands are also available from the toolbar. WFN is disabled by default.

Using nmfe

Execute a model using *Run (nmfe)*, or press Control-R. This will open up a dialog showing two additional options for running the model, e.g. which NONMEM installation to use, and if models should be executed in separate folders or on a cluster system.

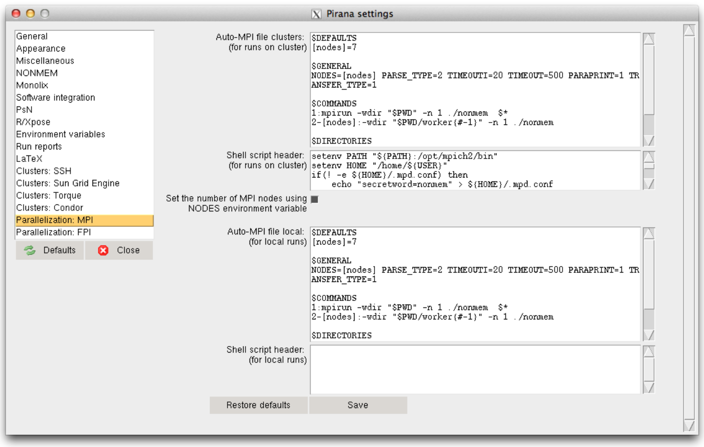


Model file(s): run2
NONMEM: nm_7.2.0_g
Run directory: /Users/ronkeizer/data/Examples2
[✓ Run in separate folder(s)]
Run multiple models in parallel
Run in background
Clusters: None (local) | No job-scheduler
Parallelization file: off | NA
[✓ Close this dialog window after starting run]

Figure 3.4: nmfe run window

Pirana supports the use of the parallelization functionality available in NONMEM 7.2+. When you select a NONMEM installation in the nmfe run window, the available parallization files (*parafiles*) are displayed under the parallelization tab. *Parafiles* can be generated using the Wizard in Pirana. You can also have Pirana generate the *parafile* on-the-fly (select *auto-MPI* or *auto-FPI*). Under Settings → Parallelization, the FPI and MPI files that Pirana generates can be specified.

Parallization files can be imported from local or remote locations. Local import can be performed through Settings → NONMEM. Remote parallelization files can be imported from a cluster location defined under Settings → SSH.



Auto-MPI file clusters: (for runs on cluster)

```
$DEFAULTS
[nodes]=7

$GENERAL
NODES=[nodes] PARSE_TYPE=2 TIMEOUTI=20 TIMEOUT=500 PARAPRINT=1 TR
ANSFER_TYPE=1

$COMMANDS
1:mpirun -wdir "$PWD" -n 1 ./nonmem  $*
2-[nodes]:-wdir "$PWD/worker{#-1}" -n 1 ./nonmem

$DIRECTORIES
```

Shell script header: (for runs on cluster)

```
setenv PATH "${PATH}:/opt/mpich2/bin"
setenv HOME "/home/${USER}"
if(! -e ${HOME}/.mpd.conf) then
    echo "secretword=nonmem" > ${HOME}/.mpd.conf
```

Set the number of MPI nodes using NODES environment variable ■

Auto-MPI file local: (for local runs)

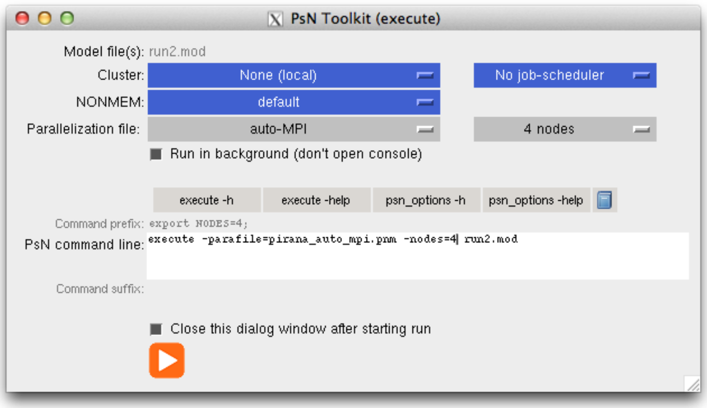
```
$DEFAULTS
[nodes]=7

$GENERAL
NODES=[nodes] PARSE_TYPE=2 TIMEOUTI=20 TIMEOUT=500 PARAPRINT=1 TR
ANSFER_TYPE=1

$COMMANDS
1:mpirun -wdir "$PWD" -n 1 ./nonmem  $*
2-[nodes]:-wdir "$PWD/worker{#-1}" -n 1 ./nonmem

$DIRECTORIES
```

Shell script header: (for local runs)

Figure 3.5: Automatic parallelization file

Using PsN

PsN is an extensive toolkit for advanced modeling & simulation, and contains essential tools such as bootstrapping, visual predictive checks (vpc), stepwise covariate modeling (scm), simulation and re-estimation (sse), and many more. All PsN-toolkit functions can be accessed from Pirana using the right-click menu or the toolbar. `execute` is also conveniently available using the *Ctrl-e* keyboard shortcut. The dialog window is then opened (shown below), which can also shows the PsN-help info for the selected command. By default, Pirana will show the dialog in *simple view*, while the *advanced view* will show more options e.g. for running R scripts before/after runs, and running on clusters and interacting with job schedulers. The command line editor can be used to specify additional parameters to the PsN funtion. Pirana stores each executed PsN command, which are available from the 'History' button.



Model file(s): run2.mod
Cluster: None (local) | No job-scheduler
NONMEM: default
Parallelization file: auto-MPI | 4 nodes
Run in background (don't open console)
Command prefix: `export NODES=4;`
PsN command line:

```
execute -parafile=pirana_auto_mpi.pnm -nodes=4 run2.mod
```

Command suffix:
Close this dialog window after starting run

Figure 3.6: PsN dialog window

From the PsN tab in the Settings window you can define the default command line parameters for most PsN functions. Some of PsN's functions are not related to models, but to datasets, such as data_stats, *create_subsets* etc. These functions can be invoked from the file list on the right by selecting a file and opening the menu by right-clicking the mouse. A similar interface will be opened as for PsN's model functions.

Similar to the nmfe dialog window, the PsN dialog also offers the possibility to easily setup parallel execution. Pirana can be instructed to auto-generate the MPI/FPI file required for parallel execution. When MPI or FPI, and the number of nodes are selected, Pirana will automatically add the required PsN arguments (`-parafile` and `-nodes`).

Using Wings for NONMEM

On Windows, Pirana is capable of invoking the WFN-commands *nmgo* and *nmbs*, for run execution and bootstrapping respectively. Since WFN does not support multiple model files to be processed by its commands, when multiple models are selected, only the first model file is executed. When the WFN method is selected, two parameter specification bars will become visible. In the upper entry, run parameters can be specified, e.g. for the bootstrap: '1 100' to specify a bootstrap with 100 replicates. The lower parameter bar specifies command-line parameters used when starting WFN.bat, e.g. 'g77 std' for specifying the compiler and the NONMEM version to be used.

Note: What Pirana actually does when executing runs through WFN, is create a temporary batch-file in the current directory that starts *WFN.bat* to load the necessary environment variables, after which *nmgo* is started with the model-file and parameters specified.

3.3 Analyzing results & output

3.3.1 Main overview

After a model has been run/executed, and the folder is refreshed, Pirana will show the main results of the run in the main overview. It will show the OFV, the difference in OFV with the reference model (if specified), the number of significant digits, and some information about the estimation, i.e.:

S means a succesful minimization (as reported by NONMEM)

R means estimation ended with rounding errors

C means a succesful covariance step

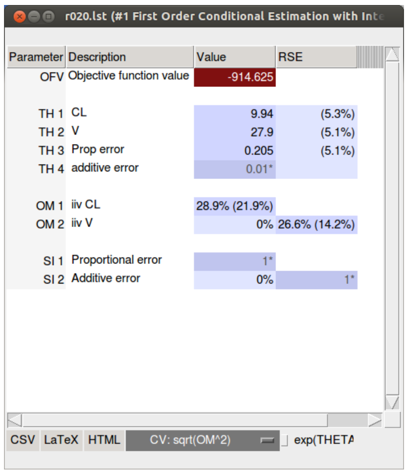
M means an unsuccesful covariance step due to matrix singularity

B means a boundary problem was reported by NONMEM

Pirana can also show the AIC and BIC values for the model, although you will have to instruct Pirana explicitly to calculate them for finished runs. See 'Miscellaneous functionality' in this manual for more information.

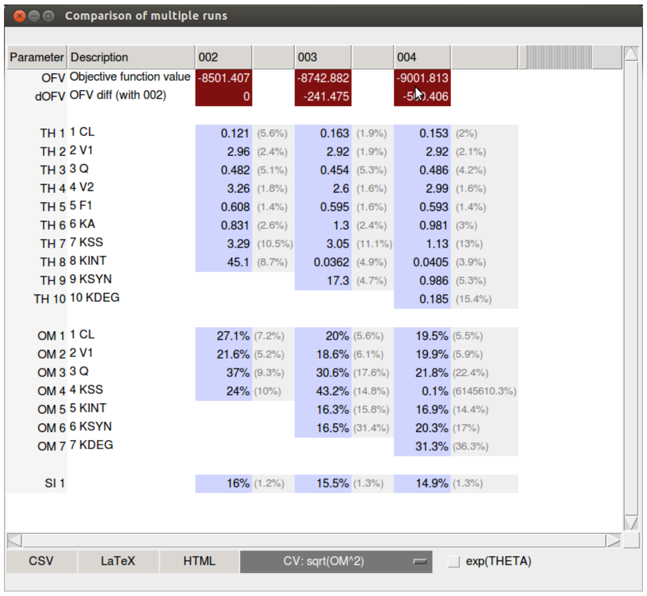
3.3.2 Parameter estimates

A list of parameter estimates is shown in the right section of the Pirana window, if *Estimates* is selected as the active tab. It also shows the RSE for parameters (between round brackets), and shrinkage for the random effects (between square brackets). In this overview it is also highlighted if final gradients for a parameter were zero (red foreground), mean of eta-distribution was significantly different from zero (etabar, red background), boundaries were encountered (blue background), or parameters were fixed (grey background). A more detailed list is available by selecting *Models* → *Parameter estimates* from the right-click menu, or from the toolbar. If just one model is selected, Pirana will show all parameter estimates and associated RSE values, if available. If multiple models/runs are selected, Pirana will show the parameter estimates of these runs side by side, facilitating comparison. These results can be easily converted into `csv`, LaTeX or `HTML` for e.g. reports or further analysis. The estimates can be exported by R as well, by clicking the R-icon in the parameter estimates area.


r020.lst (#1 First Order Conditional Estimation with Inte

Parameter	Description	Value	RSE
OFV	Objective function value	-914.625	
TH 1	CL	9.94	(5.3%)
TH 2	V	27.9	(5.1%)
TH 3	Prop error	0.205	(5.1%)
TH 4	additive error	0.01*	
OM 1	iiv CL	28.9% (21.9%)	
OM 2	iiv V	0%	26.6% (14.2%)
SI 1	Proportional error	1*	
SI 2	Additive error	0%	1*

Figure 3.7: Parameter estimates window: Single run


Comparison of multiple runs

Parameter	Description	002		003		004	
OFV	Objective function value	-8501.407		-8742.882		-9001.813	
dOFV	OFV diff (with 002)	0		-241.475		-5[illegible]0.406	
TH 1	1 CL	0.121	(5.6%)	0.163	(1.9%)	0.153	(2%)
TH 2	2 V1	2.96	(2.4%)	2.92	(1.9%)	2.92	(2.1%)
TH 3	3 Q	0.482	(5.1%)	0.454	(5.3%)	0.486	(4.2%)
TH 4	4 V2	3.26	(1.8%)	2.6	(1.6%)	2.99	(1.6%)
TH 5	5 F1	0.608	(1.4%)	0.595	(1.6%)	0.593	(1.4%)
TH 6	6 KA	0.831	(2.6%)	1.3	(2.4%)	0.981	(3%)
TH 7	7 KSS	3.29	(10.5%)	3.05	(11.1%)	1.13	(13%)
TH 8	8 KINT	45.1	(8.7%)	0.0362	(4.9%)	0.0405	(3.9%)
TH 9	9 KSYN			17.3	(4.7%)	0.986	(5.3%)
TH 10	10 KDEG					0.185	(15.4%)
OM 1	1 CL	27.1%	(7.2%)	20%	(5.6%)	19.5%	(5.5%)
OM 2	2 V1	21.6%	(5.2%)	18.6%	(6.1%)	19.9%	(5.9%)
OM 3	3 Q	37%	(9.3%)	30.6%	(17.6%)	21.8%	(22.4%)
OM 4	4 KSS	24%	(10%)	43.2%	(14.8%)	0.1%	(6145610.3%)
OM 5	5 KINT			16.3%	(15.8%)	16.9%	(14.4%)
OM 6	6 KSYN			16.5%	(31.4%)	20.3%	(17%)
OM 7	7 KDEG					31.3%	(36.3%)
SI 1		16%	(1.2%)	15.5%	(1.3%)	14.9%	(1.3%)

Figure 3.8: Parameter estimates window: Comparing multiple runs

3.3.3 Run Reports

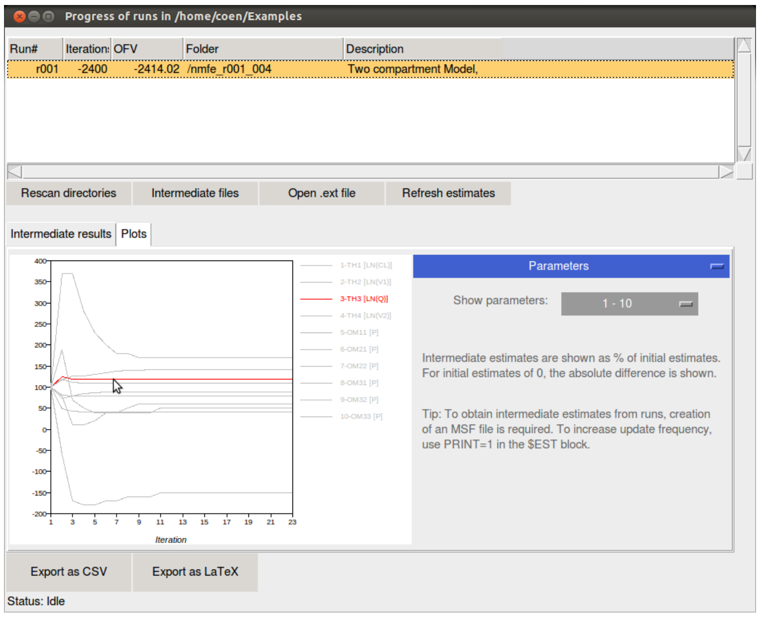
Run reports with model and run info, and parameter estimates can automatically be generated, and outputted as HTML, LaTeX, Word, or plain text format. The reports optionally displays basic run information, run statistics, description and notes, and parameter estimations, split by implemented estimation methods. The information to be included in the report can be specified in the menu under *Settings* → *Run reports*. LaTeX output is opened in the specified code editor, but also can be converted automatically to PDF using pdflatex (if installed).

After a run report is generated it will show up in the list on the right, under the tab *Reports*. In this tab, also goodness of fit plots are shown, generated either using the Xpose GUI in Pirana, or the R scripts library. Double-clicking on any of the plots or reports will re-open them.

Note: In the run reports, Pirana calculates the RSE for population parameters as $RSE_{\theta_i} = \frac{SD_{cov,\theta}}{\theta_i}$, but doesn't take into account log-transformation of parameters (e.g. when MU-modeling). For inter-individual and residual variance (Ω and Σ), RSE's are calculated as e.g. $RSE_{\omega^2_{i,i}} = \frac{SD_{cov,\omega^2_{i,i}}}{\omega^2_{i,i}}$. RSE's given for $\omega_{i,i}$ and $\sigma_{i,i}$ are calculated as e.g. $RSE_{\omega_{i,i}} = \frac{SD_{cov,\omega^2_{i,i}}}{2 \cdot \omega^2_{i,i}}$

3.3.4 Intermediate results

When running models through any of the available methods (nmfe / PsN / WFN), the intermediate results (parameter estimates, objective function value, gradients) can be viewed by clicking on the *hourglass*-icon or from *Tools* → *Intermediate results*. This will open a window which shows the models that are currently running. By clicking on a run in the list, Pirana will parse the intermediate files and show the intermediate parameter estimates in the table and the plot. In the plot, you can choose to show either the gradients (if a gradient method is used), the intermediate estimates, or the objective function value (OFV). Make sure that you specify PRINT=1 and MSFO=xxxxx in the $ESTIMATION record, to be able to obtain regularly updated intermediate estimates. From the context menu, signals can be sent to currently running NONMEM processes, such as *stop* and *next iteration*.



Run#	Iteration	OFV	Folder	Description
r001	-2400	-2414.02	/nmfe_r001_004	Two compartment Model,

Show parameters: 1 - 10

Intermediate estimates are shown as % of initial estimates. For initial estimates of 0, the absolute difference is shown.

Tip: To obtain intermediate estimates from runs, creation of an MSF file is required. To increase update frequency, use PRINT=1 in the $EST block.

Status: Idle

Figure 3.9: Intermediate results

3.3.5 Run Records

For all models in the current folder, or a selection thereof, a csv run record can be compiled by Pirana that includes all model and run characteristics, such as model description, estimation method, objective function value, termination result, etc. An abridged version of the run record can also be created as a plain text or Word document.

Visual Run Record

Pharmacometric model development most often progresses in a hierarchical fashion, using the likelihood ratio test to assess significance of improved fit between nested models. An appropriate visualization of the model hierarchy will help the modeler gain better understanding of key stages in model building, and willl aid in communicating the model development history to others. Such a visualization can be generated by Pirana, an example of which is shown below.

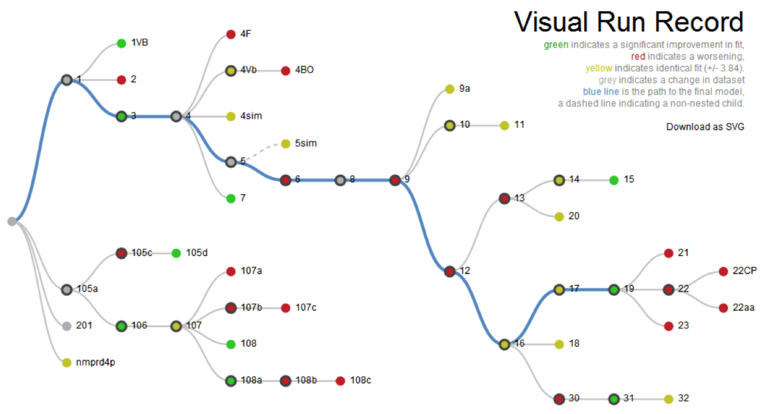
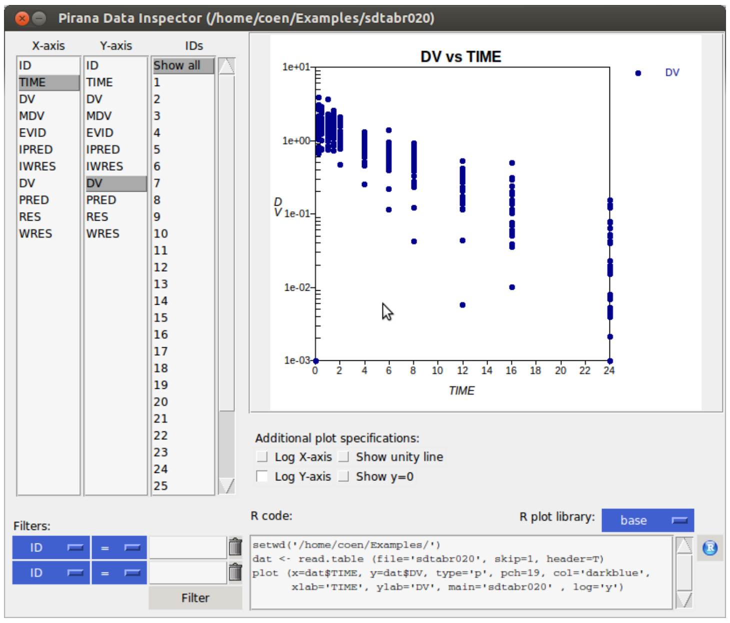

Visual Run Record
green indicates a significant improvement in fit,
red indicates a worsening,
yellow indicates identical fit (+/- 3.84),
grey indicates a change in dataset,
blue line is the path to the final model,
a dashed line indicating a non-nested child.
Download as SVG
1VB
1
2
3
4F
4
4Vb
4BO
4sim
5
5sim
6
7
8
9
9a
10
11
12
13
14
15
20
16
17
19
21
22
22CP
22aa
23
18
30
31
32
105c
105d
105a
201
106
107
107a
107b
107c
108
108a
108b
108c
nmprd4p

Figure 3.10: Visual Run Record

To create the VRR, Pirana produces Javascript code compatible with the Data-Driven Documents Javascript library (d3.js). The visualization can be generated as an SVG image in any modern internet browser. Several options are implemented that allow customization of the visualization, and both dynamic and static images can be created. In a VRR, the hierarchy of models is immediately visible, and using the dynamic collapsable dendrograms, non-relevant model threads can be hidden from the visualization. The VRR also shows additional model information when hovered over the nodes. Colors aid in visualizing the improvement / worsening of model fit (green / red), and whether the model has children or not. In each branch, the nodes are ordered by OFV. When a final model is specified, the modeling path can be made visible as a blue line, thereby easily identifying the key runs. The visualization can be implemented as a horizontal or radial tree, the latter of which can be used when the model tree is very large.

3.3.6 Data Inspector

Pirana is able to construct scatter plots using the built-in *DataInspector*, e.g. for quickly inspecting goodness-of-fit, covariate relationships, distribution of etas, performing data checkout etc. The DataInspector shows all the columns present in the dataset, which can be mapped to the X- or Y-axis (shown in figure 3.11). DataInspector can either be invoked by selecting a model (main overview), or a data file (from the list on the right). If a model is selected and DataInspector is started, Pirana will gather the dataset specified in `$INPUT` and the table files created in `$TABLE`. It will then ask the user which of these files to open in DataInspector. If only one file is found, it will open that one automatically.



```
setwd('/home/coen/Examples/')
dat <- read.table (file='sdtabr020', skip=1, header=T)
plot (x=dat$TIME, y=dat$DV, type='p', pch=19, col='darkblue',
      xlab='TIME', ylab='DV', main='sdtabr020' , log='y')
```

Figure 3.11: Pirana DataInspector

3.3.7 Visual run records

The model building process from initial to final model can be visualized using the visual run record. This can be done via Results → Run records → Visual run record. Here a final model can be selected, and the visual run record will be graphically depicted.

Pirana can parse NONMEM-generated tables, and `csv` files. Multiple Y values can be plotted by holding the Control- or shift-key and selecting multiple (up to three) data columns. Inside the plot, regions of interest may be selected, which are then zoomed. Double-clicking inside the plot region changes back to the previous view. Plots can be filtered, which can be useful, e.g. to show only data for one patient, or groups of patients or covariates.

In the text-box below the plot, code is generated that recreates the same graph in R, either in *base*, *lattice*, or *ggplot2*. This code can be used as a starting point for the generation of plots for manuscripts or reports. Clicking the *R*-button will invoke the R-GUI or code editor.

3.3.8 R scripting for creating graphs and file processing

Pirana includes functionality to run custom R-scripts on output from models-executions such as NONMEM tables. Scripts can be written by the user, but a considerable collection of scripts is also bundled with Pirana, which can serve as starting point for your own implementations. Scripts are located in three locations, one for group-wide scripts (in the scripts-folder in the location where Pirana is installed), one for user-scripts (`home/user/.pirana/scripts` on Linux, `C:\Documents and Settings\user\Application data\.pirana\Scripts` on Windows), and one for project-specific scripts, stored in the subfolder `pirana_scripts` in the current folder. The folder structure underlying the scripts folders is reconstructed within the scripts menu, and scripts can be edited either by editing them from outside Pirana, or by choosing them from the *Edit* menu option.

Scripts can be started by selecting a model, and selecting the desired script from the scripts menu located in tight tab panel. Pirana invokes R and runs the script in the directory `pirana_temp` underlying the active folder. However, before execution, Pirana replaces `#PIRANA_IN` with an R list-object which specifies model and results information, e.g. as:

```
models <- list (
   "003" = list (
     "modelfile"        = "003.mod",
     "description"      = "PK model digoxin",
     "reference_model" = "002",
     "data_file"        = "nm_pk_001.csv",
     "output_file"      = "003.lst",
     "tables"           = c("003.TAB", "sdtab003")
   )
 )
```

To automatically load PDFs or images that are created by R after execution of the script, you should include e.g. the following line in the script:

```
#PIRANA model_output.pdf
```

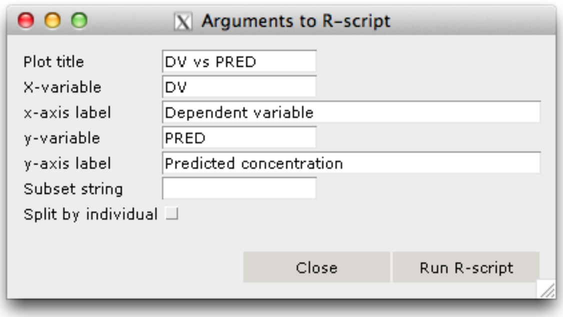
where `model_output.pdf` is the file that you want Pirana to load. This may either be a PDF, or a common image format such as `png`, `jpg`, or `gif`. Please have a look at the scripts included with Pirana for examples this functionality.

Interactive scripts

Pirana also has the ability to create *interactive scripts*, meaning that upon execution of an R-script, the user will be presented with a dialog that asks for plotting and input options. The plotting options can be specified in the R-script like e.g.:

```
### <arguments>
###       <title label="Plot title">DV vs PRED</title>
###       <x_var label="X-variable">DV</x_var>
###       <xlab label="x-axis label">Dependent variable</xlab>
###        <y_var label="y-variable">PRED</y_var>
###        <ylab label="y-axis label">Pred. concentration</ylab>
###        <subset label="Subset string"></subset>
###        <split_id label="by ID" type="bool">FALSE</split_id>
### </arguments>
```

This will result in the following interface:



Plot title	DV vs PRED
X-variable	DV
x-axis label	Dependent variable
y-variable	PRED
y-axis label	Predicted concentration
Subset string	
Split by individual	☐

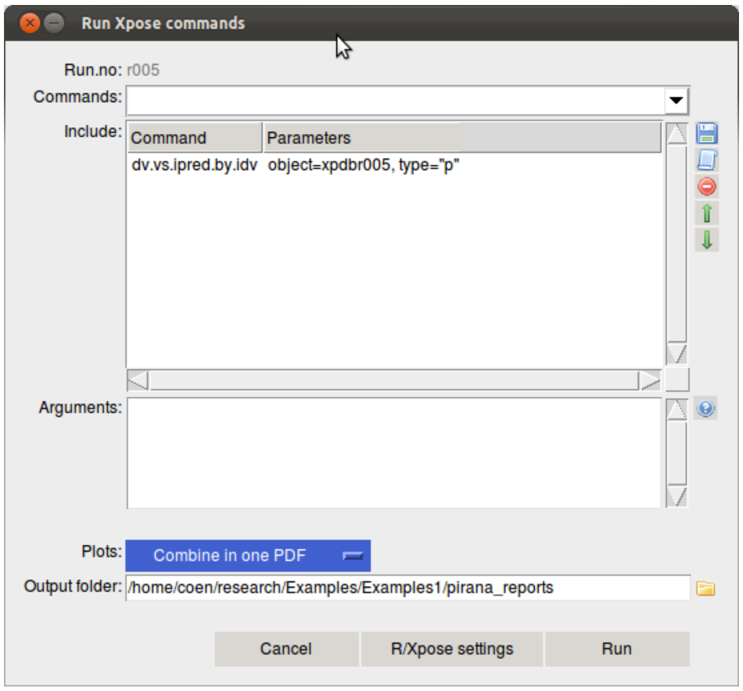
Figure 3.12: Interactive scripts

In the R-script, the specified options are then available as the list *arg*, e.g using:

```
ggplot (data=tab, aes (x=get(arg$x_var),
                       y=get(arg$y_var))) + geom_point()
```

3.3.9 Xpose support

After selecting a model, Xpose can be invoked from the scripts menu either by selecting the integrated Xpose GUI or the calling the (external) Xpose R-menu. The integrated Xpose GUI allows for execution of Xpose commands. The user can also add optional arguments to Xpose commands. Plots can be saved as PDF or PNG files, or Sweave/knitr code can be generated.


Run Xpose commands

Run.no: r005

Commands:

Include:

Command	Parameters
dv.vs.ipred.by.idv	object=xpdbr005, type="p"

Arguments:

Plots: Combine in one PDF

Output folder: /home/coen/research/Examples/Examples1/pirana_reports

Figure 3.13: Xpose window

Besides the commands available in the list, you can also input your own commands or statements to the list. Lists can be saved for easy access later on. The general plotting arguments for `pdf` and `png` (e.g. `width=10, height=8` can be set in the settings menu under *R/Xpose*)

3.3.10 Wizards

As mentioned earlier, Pirana comes with several wizards, such as a wizard to create a NONMEM model file, PsN-scm configuration files, and parallelization files. You can alter these Wizards to your liking, as they are implemented from wizard-specification files located in `/wizards` in the Pirana installation folder. Of course, it is possible to create your own Wizards as well. Just create a `.pwiz` file in the wizards-folder and follow the instructions in the template that is provided there. Wizards that are included with the current version of Pirana are described below:

PK model

This wizard allows stepwise construction of a range of PK models in NONMEM. It includes all ADVANs in NONMEM, all estimation methods, and the most commonly used residual error models. Of course, keep in mind that you have to change the initial estimates and the `$DATA` and `$INPUT` records to suit your PK problem.

NM parallelization file

`NONMEM 7.2` and higher allow parallization of single runs, which requires a so-called `parafile`, a configuration file for the parallelization. These files can be created using this wizard.

SCM configuration file (PsN)

The `scm` command in PsN requires a configuration file. With this wizard you can create such a file, which includes the most commonly used options. Please note that more features are available in the scm tool than are offered as option in the Wizard, so it is advised to acquaint yourself with the full scm documentation.

Dataset template

This wizard can be used to create an R script that, in turn, generates a NONMEM simulation data file with specified number of individuals, doses, observations, dosing times, and covariates. This can be useful e.g. for quickly setting up simulations in NONMEM.

3.4 Model translation

Pirana can translate NM-TRAN models written in any ADVAN routine to ordinary differential equations (ODEs). The model can be exported to ADVAN6 (NM-TRAN), to R (using the deSolve library), Berkeley Madonna, MATLAB and PopED. Also several translators are included that translate specific parts of NONMEM code to alternate NONMEM code.

Translation to other tools

For R, a multidose simulation is automatically implemented, for the other converters only single dose simulation is implemented. Pirana currently does not read in the dataset to extract dosing information.

Pirana extracts the parameter estimates for the structural model (θ), and also the between subject variability matrix (Ω). The latter is however done only when simulating in R or Berkeley Madonna (not available for Matlab at current). No residual error model is currently implemented in any of the translators, but this can easily be added by the user.

Porting the model structure to PopED allows evaluation of optimal study designs (OD). Pirana creates the necessary files for PopED execution, however the user still needs to provider the details of the design and other optimization settings.

MU-model conversion

Pirana can convert standard NONMEM models to MU-model coding. At current, Pirana can only convert models written using normal- or log-normal ηs, e.g.

```
CL = THETA(1) * EXP(ETA(1))
```

will be converted into:

```
MU_1 = LOG(THETA(1))  ; ** MU-referenced by Pirana
CL = EXP(MU_1+ETA(1))
; Original equation: CL = THETA(1) * EXP(ETA(1))
```

Difference equations

For some models written in ODEs, writing some parts of the model in difference equations can considerably reduce computational burden, while maintaining parameter precision.[1] Pirana can help in setting this up: it will transport all code written in `$DES` other than the $\frac{dA}{dt}$ system to `$PK`, and adds some required code (using *MTIME*).

3.5 Batch operations

Pirana offers functionality to perform batch operations on a set of models, such as search and replace functions.

Search and replace in models Replaces a given search text with another string or block of text in the selected models.

Replace block This function enables you to replace a whole block of code in selected model files, e.g. the `$DATA` block if you want all model files to use a different data file, or the `$THETA` block if you want to use other initial estimates.

Add code to models With this function, lines of code can be added to the beginning or the end of selected models.

Add code to blocks With this function, lines of code can be added to a specific block in the selected models.

[1] Petersson KJ et al. J Pharmacokinet Pharmacodyn. 2010 Oct;37(5):493-506.

Batch duplicate Creates multiple duplicates of one model file, with (optionally) updated run/table numbers and final parameter estimates.

Random simulation seeds In all selected models, the `$SIMULATION` block will be updated with new seeds.

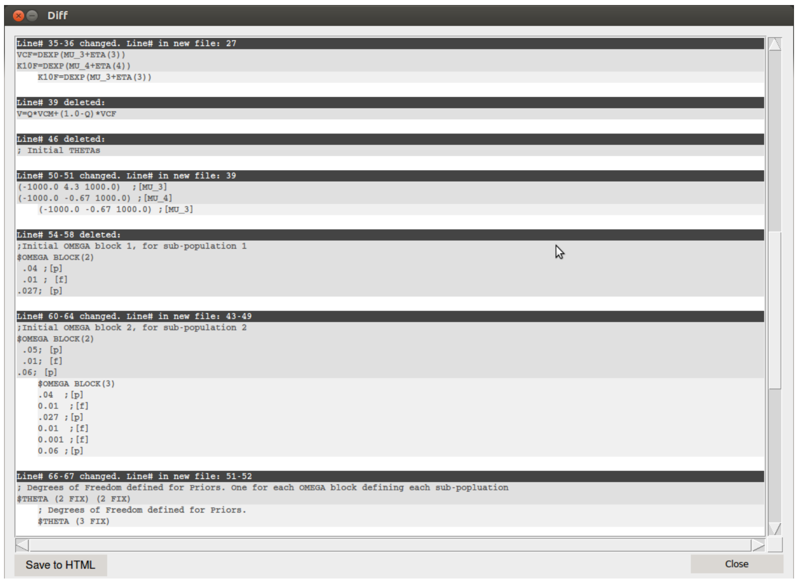
3.6 Miscellaneous functionality

NONMEM help files

Modeling with NONMEM, and construction of NM-TRAN syntax can be troublesome at times, and therefore it is convenient to have NONMEM's help files at your fingertips. Pirana provides a user interface to these help files, allowing you to filter on keyword. Before help file interface can be used, the NONMEM help files need to be imported as the files are not supplied with Pirana. For this, go to *Tools* → *NONMEM* → *Import / update NONMEM help files*. You can import the information from a local NONMEM installation or from an installation located on a cluster (over SSH). After succesful import, the NONMEM help interface is availabe from the *Help* menu.

Code differences between models

Pirana provides a tool to show code differences, similar to the *diff* functionality on Unix systems. If one model is selected and the diff tool is activated (*Right-click menu* → *Model* → *Code difference between models*), Pirana will show the difference between that model and the reference mode (if specified). If two models are selected, Pirana will show the code differences between these two models.

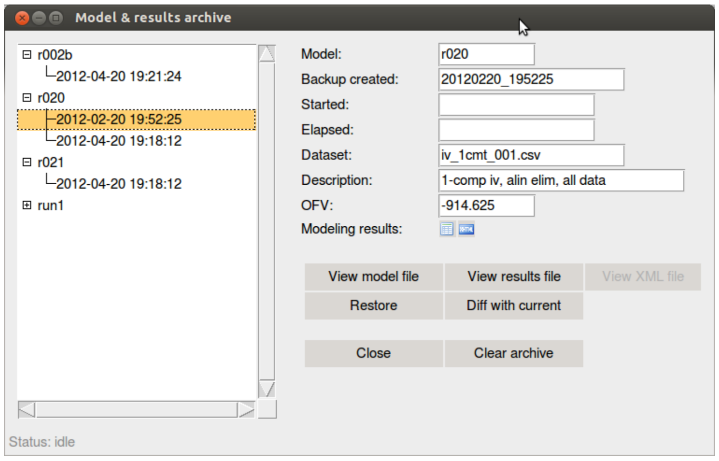


```
Line# 35-36 changed. Line# in new file: 27
VCF=DEXP(MU_3+ETA(3))
K10F=DEXP(MU_4+ETA(4))
    K10F=DEXP(MU_3+ETA(3))

Line# 39 deleted:
V=Q*VCM+(1.0-Q)*VCF

Line# 46 deleted:
; Initial THETAs

Line# 50-51 changed. Line# in new file: 39
(-1000.0 4.3 1000.0)  ;[MU_3]
(-1000.0 -0.67 1000.0) ;[MU_4]
    (-1000.0 -0.67 1000.0) ;[MU_3]

Line# 54-58 deleted:
;Initial OMEGA block 1, for sub-population 1
$OMEGA BLOCK(2)
 .04 ;[p]
 .01 ;  [f]
.027;  [p]

Line# 60-64 changed. Line# in new file: 43-49
;Initial OMEGA block 2, for sub-population 2
$OMEGA BLOCK(2)
 .05;  [p]
 .01;  [f]
.06;  [p]
    $OMEGA BLOCK(3)
    .04   ;[p]
    0.01   ;[f]
    .027 ;[p]
    0.01   ;[f]
    0.001 ;[f]
    0.06 ;[p]

Line# 66-67 changed. Line# in new file: 51-52
; Degrees of Freedom defined for Priors. One for each OMEGA block defining each sub-popluation
$THETA (2 FIX) (2 FIX)
    ; Degrees of Freedom defined for Priors.
    $THETA (3 FIX)
```

Figure 3.14: Code difference between models

Model Archive: Automatic backup of models and results

When editing and running models, Pirana can automatically backup all versions of controlstreams and result files. When this feature is activated (in the settings menu), intermediate versions of models and results are saved in a separate folder, every time a model is changed, or when a new results files is found. The archive can be accessed via the *Tools* menu, under the option *Model Archive*. Details of the run and (if applicable) parameter estimates can be reviewed. Also, different versions of models can be compared and restored. The internal Pirana database files (`pirana.dir`) are also backup up, if it has been more than a week since the previous backup.



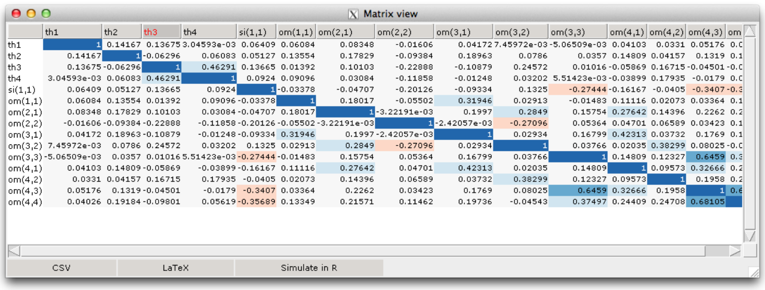
Model:	r020
Backup created:	20120220_195225
Started:	
Elapsed:	
Dataset:	iv_1cmt_001.csv
Description:	1-comp iv, alin elim, all data
OFV:	-914.625
Modeling results:	

Status: idle

Figure 3.15: Model backup / archiving window

Matrices

Pirana can automatically extract the covariance, correlation and inverse covariance matrices from a NONMEM 7+ run (`cor`/`cov`/`coi` files), and show them in a spreadsheet-like window. These can then also be automatically exported to an R object, e.g. useful for simulation purposes.



	th1	th2	th3	th4	si(1,1)	om(1,1)	om(2,1)	om(2,2)	om(3,1)	om(3,2)	om(3,3)	om(4,1)	om(4,2)	om(4,3)	om
th1	1	0.14167	0.13675	3.04593e-03	0.06409	0.06084	0.08348	-0.01606	0.04172	7.45972e-03	-5.06509e-03	0.04103	0.0331	0.05176	0.0
th2	0.14167	1	-0.06296	0.06083	0.05127	0.13554	0.17829	-0.09384	0.18963	0.0786	0.0357	0.14809	0.04157	0.1319	0.1
th3	0.13675	-0.06296	1	0.46291	0.13665	0.01392	0.10103	-0.22888	-0.10879	0.24572	0.01016	-0.05869	0.16715	-0.04501	-0.0
th4	3.04593e-03	0.06083	0.46291	1	0.0924	0.09096	0.03084	-0.11858	-0.01248	0.03202	5.51423e-03	-0.03899	0.17935	-0.0179	0.0
si(1,1)	0.06409	0.05127	0.13665	0.0924	1	-0.03378	-0.04707	-0.20126	-0.09334	0.1325	-0.27444	-0.16167	-0.0405	-0.3407	-0.3
om(1,1)	0.06084	0.13554	0.01392	0.09096	-0.03378	1	0.18017	-0.05502	0.31946	0.02913	-0.01483	0.11116	0.02073	0.03364	0.1
om(2,1)	0.08348	0.17829	0.10103	0.03084	-0.04707	0.18017	1	-3.22191e-03	0.1997	0.2849	0.15754	0.27642	0.14396	0.2262	0.2
om(2,2)	-0.01606	-0.09384	-0.22888	-0.11858	-0.20126	-0.05502	-3.22191e-03	1	-2.42057e-03	-0.27096	0.05364	0.04701	0.06589	0.03423	0.1
om(3,1)	0.04172	0.18963	-0.10879	-0.01248	-0.09334	0.31946	0.1997	-2.42057e-03	1	0.02934	0.16799	0.42313	0.03732	0.1769	0.1
om(3,2)	7.45972e-03	0.0786	0.24572	0.03202	0.1325	0.02913	0.2849	-0.27096	0.02934	1	0.03766	0.02035	0.38299	0.08025	-0.0
om(3,3)	-5.06509e-03	0.0357	0.01016	5.51423e-03	-0.27444	-0.01483	0.15754	0.05364	0.16799	0.03766	1	0.14809	0.12327	0.6459	0.3
om(4,1)	0.04103	0.14809	-0.05869	-0.03899	-0.16167	0.11116	0.27642	0.04701	0.42313	0.02035	0.14809	1	0.09573	0.32666	0.2
om(4,2)	0.0331	0.04157	0.16715	0.17935	-0.0405	0.02073	0.14396	0.06589	0.03732	0.38299	0.12327	0.09573	1	0.1958	0.2
om(4,3)	0.05176	0.1319	-0.04501	-0.0179	-0.3407	0.03364	0.2262	0.03423	0.1769	0.08025	0.6459	0.32666	0.1958	1	0.6
om(4,4)	0.04026	0.19184	-0.09801	0.05619	-0.35689	0.13349	0.21571	0.11462	0.19736	-0.04543	0.37497	0.24409	0.24708	0.68105	

Figure 3.16: Correlation matrix

Miscellaneous tools

Correlation calculator This opens the built-in *Correlation Calculator*, which can be used to re-calculate a covariance to a correlation on the SD-scale. The formula for correlation that is used is:

$$\rho_{i,j} = \frac{{\omega_{i,j}}^2}{\omega_{i,i} \cdot \omega_{j,j}}$$

with ρ specifying the correlation between two elements (i,j) in a matrix, and ω specifying elements of Ω or Σ.

Checkout dataset This will create (and open) an HTML-file which displays a selected dataset using separate colors for different event-types. This will thus show the dataset in a slightly more convenient format for manual inspection than in a spreadsheet. The function needs at least the `ID`, `TIME` and `EVID` columns in the dataset to work properly.

Calculation of AIC/BIC Pirana can calculate the Akaike Information Criterion and the Bayesian Information Criterion. These criterions are defined as follows:

$$AIC = 2 \cdot k - 2 \cdot ln(L) \tag{3.1}$$

$$BIC = -2 \cdot ln(L) + k \cdot ln(n) \tag{3.2}$$

with k the number of parameters in the model, L the maximized value of the likelihood function, and n the number of observations in the dataset used in fitting the model. The calculation of these criterions is however not so straightforward for non-linear mixed-effects models, and the weights/penalties applied to parts of the equation can be different in different circumstances. Pirana allows the penalties to be changed when it calculates the AIC/BIC. From the right-click menu, select *Model actions* $\rightarrow$ *Calculate AIC/BIC*, and a dialog window will open which will present the options available. Some useful literature is listed below.

- Vaida and Blanchard, Conditional Akaike information for mixed-effects models. Biometrika, 2005. 92(2): p. 351-370.
- Liang, et al., A note on conditional aic for linear mixed-effects models. Biometrika, 2008. 95(3): p. 773-778.
- Hodges and Sargent, Counting degrees of freedom in hierarchical and other richly]parameterised models. Biometrika (2001) 88 (2): 367-379.
- Donohue et al., Conditional Akaike information under generalized linear and proportional hazards mixed models, Biometrika (2011) 98 (3): 685-700.
- Delattre et al., BIC selection procedures in mixed effcts models http://hal.inria.fr/docs/00/69/64/35/PDF/RR-7948.pdf

Clean-up folder This tool can be used to clean-up files that NONMEM generates when running a model (e.g `INTER`, `FSTREAM`, etc.). So if you run a model in the main model folder (which is not considered good modeling practice), you can use this tool to clean up after model execution.

3.7 Conventions and Methods

Model file conventions

- Users of PsN and Xpose likely follow the 'Uppsala convention' of having model files named like *run1.mod*, *run2.mod*, etc. This is recommended for Pirana users as well, although Pirana is flexible in this respect. Note that Pirana removes the `run` from the modelfile name in the model overview.

- Pirana looks for a description of the model in the first part of the model file. It adheres to PsN's run record standards. If the PsN run record is not used, Pirana searches for the words `$PROBLEM` or `Model desc:` to extract the model description.

- If you want to use the hierarchy functionality for models, you should specify the reference model in the first few lines of the model file. Again, best is to use PsN's run record specification, but Pirana is flexible and also compatible with Census, and understands the following syntaxes:

```
;; 1. Based on: 001.mod
; Ref. model:001.mod
; Ref:001.mod
; Parent=001.mod
```

- Model parameter descriptions need to be specified after a semi-colon, e.g.

```
$THETA
(3, 5, 11) ; CL/F
(10, 50, 100) ; V/F
```

  Note that Pirana reads these decriptions from the model file (and not from the output file). To be read correctly, covariance block need to be specified as:

```
$OMEGA BLOCK(2)
0.1 ; IIV CL/F
0.05 ; COV CL~V
0.1 ; IIV V/F
```

  or as:

```
$OMEGA BLOCK(3)
0.1           ; IIV CL/F
0.05 0.1      ; IIV V/F
0.01 0.05 0.1 ; IIV KA
```

- When models are to be executed in a separate directory, files needed for compilation (e.g. additional Fortran routines in `.FOR files`), are copied automatically by Pirana. These files should be specified in the `OTHER` and `CONTR` entries on the `$DATA` record. If more additional files are needed, you can instruct Pirana to copy these by adding this line to your control stream:

  ```
  ; INCLUDE=file1_to_be_copied.ext,file2_to_be_copied.ext,
    etc
  ```

  Note that PsN has it's own functionality for doing this.

3.8 Keyboard shortcuts

The following keyboard shortcurts are available in Pirana:

- Ctrl-R: Run model
- Ctrl-L: Open NM output file (.lst)
- Ctrl-N: New model file
- Ctrl-D: Duplicate model file
- Ctrl-P: Show parameter estimates for run(s)
- Ctrl-T: HTML-file from NM output
- Ctrl-E: Execute model (PsN)
- Ctrl-B: Bootstrap model (PsN)
- Ctrl-V: VPC from folder (PsN)
- Ctrl-U: Update inits (PsN)
- Ctrl-X: Run Xpose commands
- Ctrl-A: Select all models
- Ctrl-+/-: Increase / decrease font size
- Ctrl-, : Open settings window
- F5 : Refresh current folder

3.9 Command line parameters

- `-portable`
  Use Pirana in portable mode, e.g. from a USB-stick, leave no footprint on computer.
- `-console`
  Leave console window open. This may be useful when Pirana hangs or crashes, as sometimes an error may be shown on the command line. (Fortunately Pirana hardly ever crashes, though).
- `-debug`
  Print information during startup. This may be useful when for some reason Pirana doesn't start up, possibly due to some missing file or incomplete installation.
- `-dev`
  Pirana will show and allow use of experimental features (experimental features differ between versions).
- `-time`
  Print some information during folder reading. Useful for troubleshooting on slow network connections.

3.10 Troubleshooting / FAQ

Below are some answers to commonly asked questions. Note that a more elaborate and up-to-date FAQ is also available on the website.

- *Why the name Pirana?*

  An acronym for: Pirana is a Resourceful Assistant in NONMEM Analysis.

- I'm running on Windows, but for some reason Pirana won't start (anymore)

  Please check for any errors during startup by using the `-console -debug` arguments on the command line. If there is any problem during initialization of Pirana, you can try to reset the default settings in Pirana by running `pirana.exe -refresh` in the console (this works only in version 2.5.1 or later. For earlier versions, remove the folder `%HOME%/.pirana` manually, where `%HOME%` is your home folder). This will remove your current settings and re-install the defaults. Then, try to restart Pirana in the regular way. If this doesn't work, please report it as a bug with a screendump of the console output.

- *In Pirana's model overview table, I see some models but no results...?*

  First check if you have the extensions set correctly in preferences, e.g. .mod/.lst for model/results files. If this doesn't solve the problem, it may be due to the fact that your database file for that folder is corrupt. In the current folder, try renaming pirana.dir to pirana.dir.bak and refresh the folder. This error may sometimes occur when you have been using an old version of Pirana previously (<2.1.0). On Linux, problems have been reported with old versions of the Perl modules `DBI` and `DBD::SQLite`. Make sure you have versions 1.6 or higher or 1.14 or higher, of these respective modules. (You can check this by typing the following on the command line:

  ```
  perl -MDBI -e 'print "$DBI::VERSION"'
  perl -MDBD::SQLite -e 'print "$DBD::SQLite::VERSION"'
  ```

- *On startup of Pirana on Windows, it complains that it can't find the file* `libgcc_s_dw2/1.dll`, `libstdc++-6.dll`, *or* `perl516.dll`. Pirana needs these library files to function, and therefore we supply them with Pirana (in the folder where you installed it, probably C:\Program Files\Pirana). On some systems however, it needs to be copied to the folder "C:\Windows" or "C:\Windows\system32" for Pirana to function.

- *I can't seem to start any NONMEM runs or use PsN. Basically, noothing happens, and no error message is produced.*

On Windows: The likely cause of this error is that Pirana can't start your command shell. Please check in your "Environment variables" in the Control panel, that

```
C:\Windows\system32
```

is included in the `PATH` environment variable. If not, add it. While you're at it, check also that the variable ComSpec is set to:

```
C:\Windows\system32\cmd.exe
```

on Mac OSX: This is most likely because Pirana is not able to find X-terminal. Please update your settings in Pirana, in *Settings* → *Miscellaneous* → *Terminal to start NONMEM runs in*. Change `xterm` into `/usr/X11/bin/xterm`, and it should work fine.

- *When I open a csv file in Excel from Pirana (and in general), I get the error message that the file is recognized as a SLYK-type file, and therefore cannot open the file.*

  This is due to the fact that the first column in your dataset is named `ID`. Renaming the column (e.g. `id`) or inserting a column before it will solve the problem. When converting and opening table files, the `ID` column is automatically converted to `id` by Pirana to avoid these issues.

- *Running on Mac OS X, after starting Pirana, nothing seems to happen.*

  Make sure you have Xcode, the Xcode command line tools, and XQuartz installed. Update to the latests OS X version.

- *Where does Pirana store the notes I make in the model overview?*

  Pirana stores your notes in a database-file (`pirana.dir`) in the current folder. So, if you would install a new version of Pirana, or move your model folder to another place, your notes will not be lost. The database file contains also some other information about the run results (OFV, run success, etc.) and can of course be read out manually as well, using `sqlite3`.

- *I'd like to cite Pirana in a report or article.*

  Please cite Pirana as: Keizer RJ et al.; Comput Methods Programs Biomed 2011 Jan;101(1):72-9; Piraña and PCluster: a modeling environment and cluster infrastructure for NONMEM. PubMed

3.11 Monolix

Since version 2.7.1, Pirana includes support for Monolix, the SAEM nonlinear mixed-effects modeling system developed by Lixoft. Pirana supports both the stand-alone version and the Matlab version of Monolix, and interacts with Monolix both on Windows and on Linux. Pirana requires that models are written in MLX-TRAN, the modeling language derived from NM-TRAN that can be used to write models in Monolix. Similar to NONMEM models, MLX-TRAN models are shown in the main overview automatically when a folder is loaded in Pirana. Pirana will automatically switch to *Monolix mode* when it detects MLX-TRAN models in the folder (the extension of which can be set under *Settings*). MLX-TRAN can be run from the Monolix submenu, and results and parameter estimates can be viewed in a similar fashion as the results obtained from NONMEM.

Note: The Monolix functionality in Pirana is still in development, and not all features available for NONMEM are available for Monolix. Also, a more detailed description of available features will be added to a future version of this manual. If you are interested to have Monolix-specific features implemented in Pirana, please let us know.

Chapter 4

Pirana and clusters

4.1 Clusters

Pirana supports interaction with Linux-based clusters on which NONMEM and/or PsN are installed. The Job-schedulers Sun Grid Engine (SGE), Torque and Condor are supported. In addition, SSI-type cluster managers such as MOSIX should also work without problems. Connecting to a cluster is established using the SSH protocol, or any method that can be invoked from the command line. Two methods are available for using Pirana with a grid/cluster system, which involve installation of Pirana either on the local system or directly on the cluster server. The following paragraphs discuss these two separate methods.

Note: Single-system image clusters such as MOSIX, openMOSIX and Kerrighed distribute processes automatically accross nodes, and therefore no alternative setup is required in Pirana.

4.1.1 Method 1: Server-based installation

When using this approach, Pirana is only installed on the cluster-server, not on the local machine. Pirana is executed from the local machine using *X-over-SSH* window tunneling. This has the advantage of requiring only one central installation of Pirana for the entire modeling group, and Pirana and other modeling software is installed in a controllable environment. A disadvantage is that the interface is usually a bit slower. Also all auxiliary software (Office suite, HTML-browser, R and an R-GUI, etc.) resides on the cluster. Especially when using this method over larger distances (i.e. across internet), the performance of Pirana may be impaired due to the server-client transmission of the full GUI, but this of course depends on the bandwith of your connection and can be tested easily.

X-over-SSH tunneling

On the local machine it is necessary to have an X window system installed. For Linux users this is likely already installed. Mac OSX users need to install the XQuartz system. For Windows, a good X window manager is *Xming*, which can be obtained for free from `http://sourceforge.net/projects/xming/`. After installation of Xming, start the Xming X-window server. An alternative to *Xming* is *Cygwin/X*.

Using the cluster

If everything is set up correctly, and the X-window server is started, Pirana on the cluster can be accessed through SSH, e.g. by using the SSH client. Now, you should be able to see the Pirana main window. If you get an error saying that the display cannot be started on localhost, you may have to enable X-window forwarding in OpenSSH or in PuTTY. When using PuTTY, it is essential to use the PuTTY terminal directly, and not plink.exe. The latter program tends to make Pirana crash often, probably due to terminal incompatibility. OpenSSH can also be used.

4.1.2 Method 2: Local installation

The other method is to install Pirana on the local machine, and connect to the cluster using Pirana and third-party SSH software. We usually recommended this installation approach as it offers a more stable interface (independent of network speed), and does not require installation of auxiliary software on the cluster. It will however require a few additional local installations. First, you need to mount the cluster drive with your data on your local PC (e.g. using sshfs on Linux/Mac or ExpanDrive on Windows). This can be set up e.g. using ExpanDrive to connect to the cluster through SFTP. Alternatively, if, on the remote cluster a Samba server is installed, a connection can be established by giving the following command:

```
NET USE Z: \ \server_name\<name> /user:<name> /persistent:yes
```

Both on Windows and Linux, you need to specify in the preferences the mounted remote diskspace and the local location, which is used by Pirana to translate local paths to paths on the remote cluster.
Secondly, an SSH client needs to be installed, which is probably already the case on Linux or Mac. On Windows, we have good experiences with PuTTY (`http://www.chiark.greenend.org.uk/∼sg`
and OpenSSH (download from `http://sshwindows.sourceforge.net/`).

In the *Settings* → *Clusters: SSH* menu, specify the command you use to connect to the cluster on the command line, e.g.:

```
ssh user@server.domain.ext
```

Note that Pirana needs passwordless SSH-access to the cluster, so make that you have an RSA key pair installed (explained in the next section). If you use PuTTY on Windows, you can also choose to supply the password on the command line instead, e.g.:

```
plink -l username -pw password server.domain.ext
```

although from a security perspective this is not recommended. Models can then be run in a similar fashion as explained in the section for running models locally, just select the cluster to run on from the `nmfe`- or PsN run windows.

Installing Public and private authentication keys

Either on Windows or Linux, type in a shell/console window: (If you use PuTTY instead of OpenSSH, use the Keypair generator program instead.)

```
ssh-keygen -t rsa
```

When asked for a passphrase, just press <Enter>. Now a public and a private key have been created in

`c:\Documents and Settings\<Name>\.ssh}`

(Windows) or

`/home/username/.ssh`

(Linux). In you home directory on the cluster, if it doesn't exist already, create the folder '.ssh'. In this folder, create the file 'authorized_keys' (no extension) and add the contents of id_rsa.pub to that file and save it. Now you should be able to login without being asked for a password. If SSH asks you if you want to accept the cluster as valid host, accept). Keep your private key secret. In the Pirana preferences, speficy the username to connect to the cluster (ssh_login).

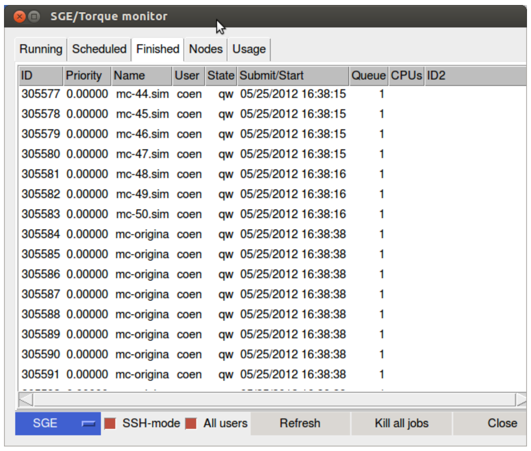
Tip: if you experience delays (about 5 secs) when logging in to the server by SSH, this may be caused by a reverse DNS lookup. You can circumvent this by adding 'useDNS no' to the file /etc/ssh/sshd_config on the server. Restart the ssh server for the changes to take effect: sudo /etc/init.d/ssh restart

4.2 Interaction with job schedulers

Pirana has integrated support for Sun Grid Engine (SGE), Torque, and Condor job scheduling systems. In the settings menu, clusters and job schedulers can be configured. For Torque and Condor, auxilliary scripts can also be defined in the settings menu. Jobs submitted through SGE, Condor or Torque can be monitored and managed using the integrated job monitor (via View menu or top-right icon). If PsN is used, Pirana also indirectly supports the use of additional job scheduling systems that are supported by PsN (SLURM, LSF, UD, MOSIX, LSF). No integrated run managers are currently available in Pirana for these additional cluster systems.

4.2.1 Working with the SGE

Submitting the execution of a model using nmfe to the SGE, Torque, or Condor can be done by selecting the 'Run on SGE or Torque' from the *nmfe* or *PsN* dialog windows. This submits the model using *qsub* instead of starting it directly.



ID	Priority	Name	User	State	Submit/Start	Queue	CPUs	ID2
305577	0.00000	mc-44.sim	coen	qw	05/25/2012 16:38:15	1		
305578	0.00000	mc-45.sim	coen	qw	05/25/2012 16:38:15	1		
305579	0.00000	mc-46.sim	coen	qw	05/25/2012 16:38:15	1		
305580	0.00000	mc-47.sim	coen	qw	05/25/2012 16:38:15	1		
305581	0.00000	mc-48.sim	coen	qw	05/25/2012 16:38:16	1		
305582	0.00000	mc-49.sim	coen	qw	05/25/2012 16:38:16	1		
305583	0.00000	mc-50.sim	coen	qw	05/25/2012 16:38:16	1		
305584	0.00000	mc-origina	coen	qw	05/25/2012 16:38:38	1		
305585	0.00000	mc-origina	coen	qw	05/25/2012 16:38:38	1		
305586	0.00000	mc-origina	coen	qw	05/25/2012 16:38:38	1		
305587	0.00000	mc-origina	coen	qw	05/25/2012 16:38:38	1		
305588	0.00000	mc-origina	coen	qw	05/25/2012 16:38:38	1		
305589	0.00000	mc-origina	coen	qw	05/25/2012 16:38:38	1		
305590	0.00000	mc-origina	coen	qw	05/25/2012 16:38:38	1		
305591	0.00000	mc-origina	coen	qw	05/25/2012 16:38:38	1		

Figure 4.1: Job monitor

4.3 PCluster

Earlier versions of Pirana supported the construction of a simple Windows-based clustering system (PCluster). PCluster allows set up of a cluster using e.g. PCs of colleagues, and is easy to install on Windows systems. However, this kind of setup can nowadays be considered inferior to the clusters mentioned above in terms of stability and performance. The development of PCluster has therefore been discontinued, and no active support will be provided for this feature. If you still want to try the PCluster, there is a short legacy manual available upon request.

Chapter 5

Automated modeling workflow in Pirana

5.1 Background

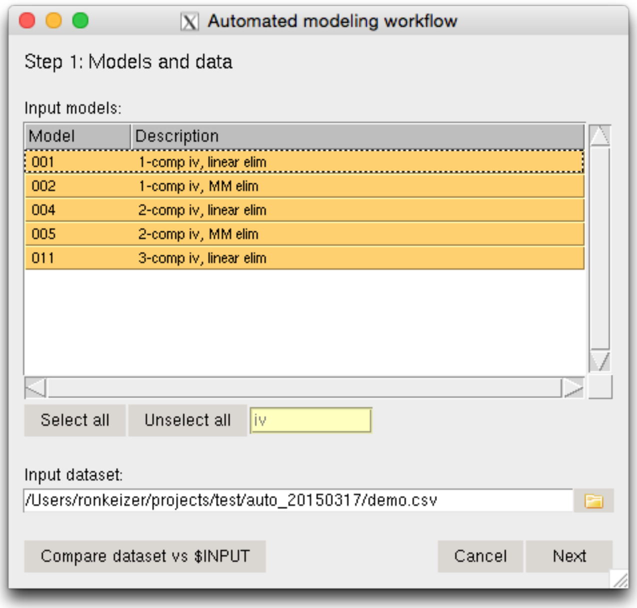
An automated workflow alleviates the burden on modeling scientist by removing the repetitive task of running and evaluating many candidate models, standardizes the model development between modelers, and standardize the results reported from such an analysis ultimately leading to higher quality of PopPK analyses (*Schmidt et al. JPKPD 2014 Aug*). In Pirana (version >= 2.10), such a workflow is made available, and in this chapter we will walk through an example of an automated population PK analysis.

5.2 Tutorial

For this chapter, we will use the template model library that is provided with Pirana, and a (simulated) dataset of an iv-administered drug also provided with Pirana (`demo.csv`).

Start

In Pirana, create a new project folder somewhere on your hard-drive (or cluster). Browse into this folder (with Pirana). In this folder, copy the file `demo.csv` that is included in the Pirana installation folder (`Pirana/automod_library/demo/demo.csv`). In Pirana, go to *Tools* → *Automated modeling workflow*.


Automated modeling workflow

Step 1: Models and data

Input models:

Model	Description
001	1-comp iv, linear elim
002	1-comp iv, MM elim
004	2-comp iv, linear elim
005	2-comp iv, MM elim
011	3-comp iv, linear elim

Input dataset:
/Users/ronkeizer/projects/test/auto_20150317/demo.csv

Figure 5.1: Screen 1: Models and dataset

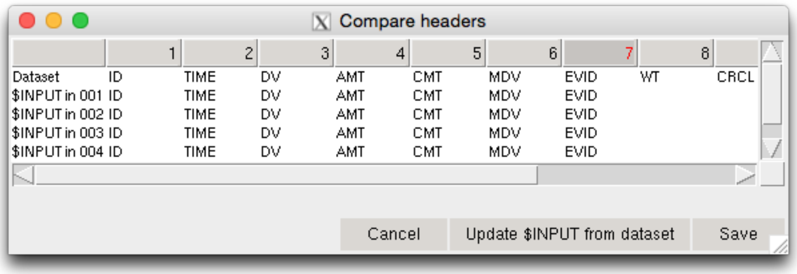
Screen 1: Models and dataset

This screen shows all models available in the library and which can be selected to be included in the analysis. Use the filter for conveniently selecting e.g. only *iv* or only *oral* models. The dataset should of course be specified as well before you can advance to the next step. By default, it will use the first `.csv` file in this folder (in alphabetic order).

When models and dataset have been selected, you should check whether the `$INPUT` record in the models matches with the headers in the dataset. For this, click the button *Compare dataset vs* `$INPUT`. This will bring up screen shown in figure 2:

If the `$INPUT` in the models (shown in rows 2-. . .) does not match up with the dataset (shown in row 1), you can click the button *Update $INPUT from dataset*. This will create a new `$INPUT` record for all models. After clicking *Save*, when the models will be written (in step 3 of the automated analysis), the `$INPUT` records in all models will be changed to the new one. It is left to the user to make sure that the variables used in the model are still included in `$INPUT`, as there is no extra check in place for that.

For our current analysis, we will select all *iv* models, so filter on *iv*, and select the remaining models. Update the `$INPUT` records, click *Save* and then *Next* to advance to



	1	2	3	4	5	6	7	8	
Dataset	ID	TIME	DV	AMT	CMT	MDV	EVID	WT	CRCL
$INPUT in 001	ID	TIME	DV	AMT	CMT	MDV	EVID		
$INPUT in 002	ID	TIME	DV	AMT	CMT	MDV	EVID		
$INPUT in 003	ID	TIME	DV	AMT	CMT	MDV	EVID		
$INPUT in 004	ID	TIME	DV	AMT	CMT	MDV	EVID		

Figure 5.2: Compare/set input headers

the next step.

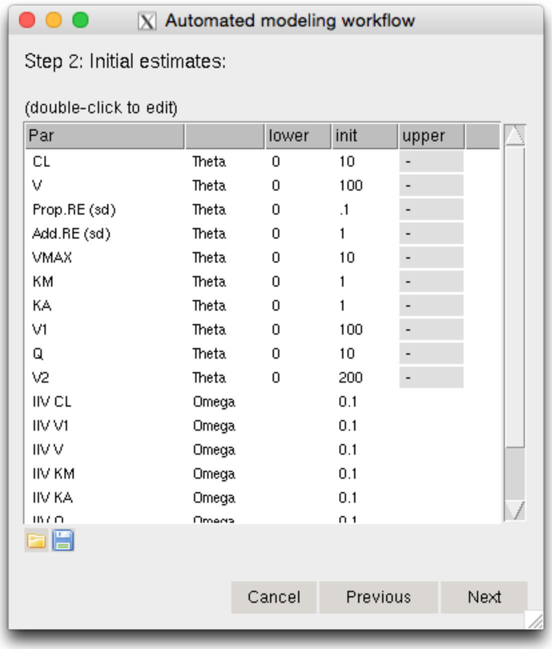
Screen 2: Setting initial parameter estimates

In the second screen, we can set initial parameter estimates, as well as lower and upper bounds. All parameters are read from the models that were selected in step 1. The parameter descriptions are defined in the models as comments to `$THETA`, `$OMEGA`, and `$SIGMA` blocks, like e.g.

```
$THETA
(0, 5, 100); CL
(0, 5, 100); V

$OMEGA
(0.1); CL
(0.1); V

$SIGMA
0.05 ; proportional error
```

Note: At current, correlations in $OMEGA and $SIGMA cannot be specified for an automated analysis. I.e. only the diagonal elements of $OMEGA and $SIGMA can be specified in the template models if you want to update them in this step. You can still include models that have full $OMEGA or $SIGMA blocks as template model, however you cannot provide descriptions (as comments) to the parameters in the block, and you cannot update them in this step of the analysis.

The two buttons below the parameter grid can be used to save and (re-)load parameter definitions to file.


Step 2: Initial estimates:

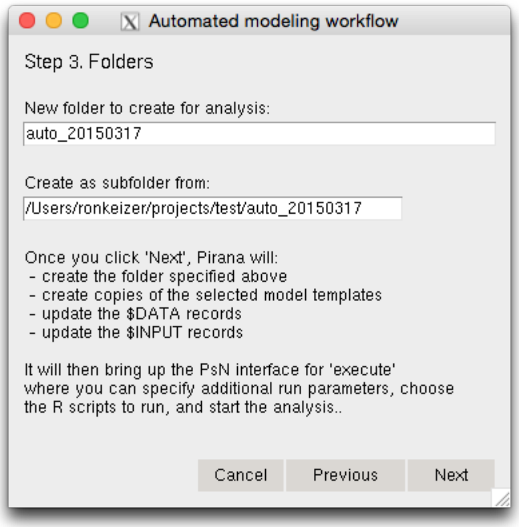
(double-click to edit)

Par		lower	init	upper
CL	Theta	0	10	-
V	Theta	0	100	-
Prop.RE (sd)	Theta	0	.1	-
Add.RE (sd)	Theta	0	1	-
VMAX	Theta	0	10	-
KM	Theta	0	1	-
KA	Theta	0	1	-
V1	Theta	0	100	-
Q	Theta	0	10	-
V2	Theta	0	200	-
IIV CL	Omega		0.1	
IIV V1	Omega		0.1	
IIV V	Omega		0.1	
IIV KM	Omega		0.1	
IIV KA	Omega		0.1	
IIV Q	Omega		0.1	

Figure 5.3: Screen 2: Initial parameter estimates

For our analysis, let's leave the parameters as they are. Click **Next** to advance to the next step.

Screen 3: Folders



Step 3. Folders

New folder to create for analysis:
auto_20150317

Create as subfolder from:
/Users/ronkeizer/projects/test/auto_20150317

Once you click 'Next', Pirana will:
- create the folder specified above
- create copies of the selected model templates
- update the $DATA records
- update the $INPUT records

It will then bring up the PsN interface for 'execute' where you can specify additional run parameters, choose the R scripts to run, and start the analysis..

Figure 5.4: Screen 3: Folders

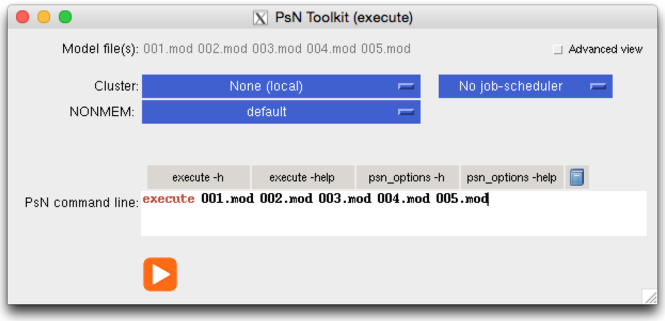
In this screen we can specify where to create the new models and run the analysis. By default it will generate a new folder name based on the current date, and as a subfolder from the current folder in Pirana. This screen also lists the actions that Pirana will perform once you click *Next*.

Let's use the defaults and click *Next*.

Screen 4: PsN dialog

Pirana should now have switched automatically to the new folder where you will see the newly generated models. Pirana will also automatically bring up the PsN `execute` dialog. In this dialog, if you switch to *Advanced view*, you can select which R script(s) to run after all runs have been completed to generate goodness-of-fit plots. Click the *folder* icons next to the R scripts textboxes to select R scripts (or batch files) to run after (or before) the analysis (figure 6).

For our analysis, we will select the *Basic GOF plots as single doc* from the `Reports`



Model file(s): 001.mod 002.mod 003.mod 004.mod 005.mod

Cluster: None (local)

No job-scheduler

NONMEM: default

PsN command line:
```
execute 001.mod 002.mod 003.mod 004.mod 005.mod
```

Figure 5.5: Screen 4: PsN

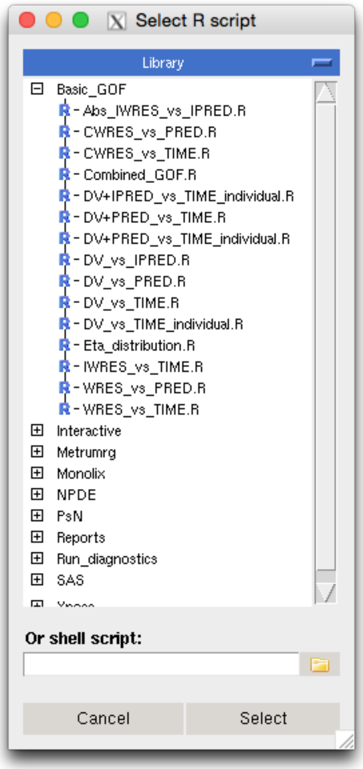
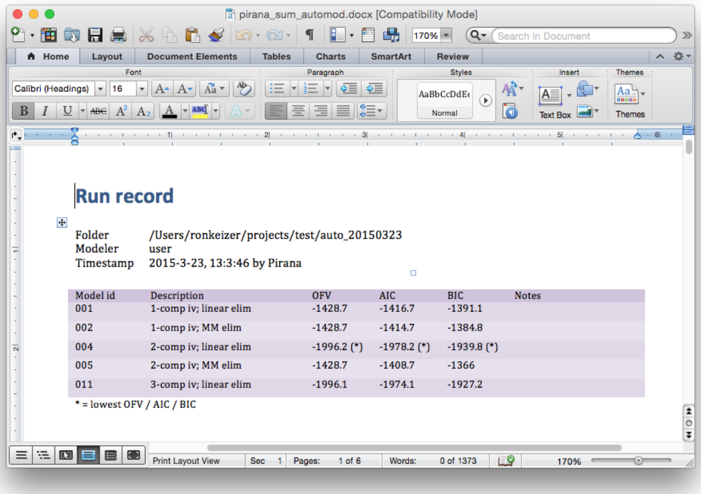


Figure 5.6: Select R script

folder to create GOF plots for all models. The graphical report will automatically be opened, but is also available from the *Reports* tab on the right.

If you have not selected R scripts to be executed automatically after the analysis has completed, you can still create them afterwards by selecting the runs and running any R script from the *R* tab on the right side of the Pirana window.

Besides the graphical report, Pirana can generate a *numeric* report for the analysis, including e.g. OFVs, basic run information and parameter estimates. This document is not generated automatically but has to be requested manually after the analysis is complete: Make sure Pirana is still in the folder where the analysis was run, and then go to *Tools* → *Automated modeling workflow* → *Report*. On the first page you will see an overview of all models included in the analysis and their respective OFV, AIC and BIC (figure 7). The subsequent pages includes information on each individual run in the analysis.



Run record

Folder	/Users/ronkeizer/projects/test/auto_20150323
Modeler	user
Timestamp	2015-3-23, 13:3:46 by Pirana

Model id	Description	OFV	AIC	BIC	Notes
001	1-comp iv; linear elim	-1428.7	-1416.7	-1391.1	
002	1-comp iv; MM elim	-1428.7	-1414.7	-1384.8	
004	2-comp iv; linear elim	-1996.2 (*)	-1978.2 (*)	-1939.8 (*)	
005	2-comp iv; MM elim	-1428.7	-1408.7	-1366	
011	3-comp iv; linear elim	-1996.1	-1974.1	-1927.2	

* = lowest OFV / AIC / BIC

Figure 5.7: Report in Word

Chapter 6

cPirana

cPirana is a simple, *lite* version of Pirana that runs on the Unix command line. There are several reasons that we think cPirana is a useful addition to our primary software product: many –usually more experienced– modelers prefer to work from the command line. However, it is our opinion that a user interface (be it graphical or console-based) can greatly increase productivity and make the modeling process easier in general. For those modelers, we hope cPirana is a welcome addition to their workflow.

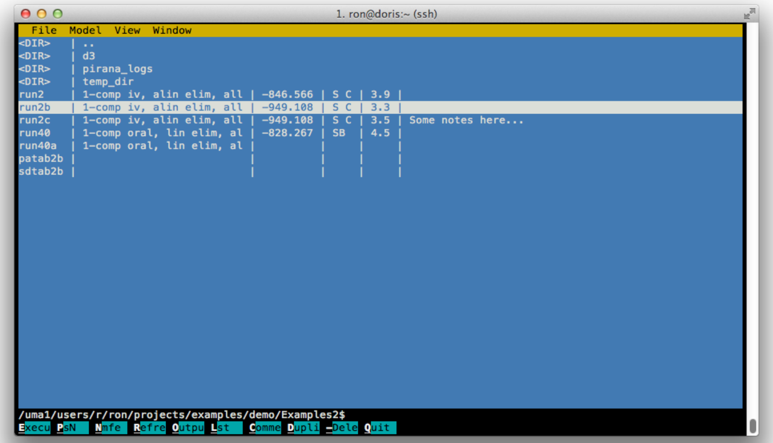
Secondly, while working on cluster through a slow internet connection, the interface of the Desktop version may become slow. Version 2.7.0 of Pirana included many speed improvements that improves working on slow connections. However, e.g. when traveling, one might be only interested in making a few small code changes and restart a model. For such use-cases, we consider cPirana to be a useful tool.

Finally, already from our earliest versions of Pirana, we intended to extend the implementation of Pirana to smartphone or tablets, allowing for increased mobility and connectivity. cPirana actually offers this possibility: if cPirana is installed on the Linux cluster that runs NONMEM and PsN, the user can connect from a tablet or smartphone using specialized ssh-apps (such as 'Prompt' on the iPad), to connect.

6.1 Installing cPirana

6.1.1 Using cPirana

Copy the contents of the linux installation zip-file file to a location on your system. Pirana is started using the command (assuming you installed it in the folder /pirana inside your home folder):



```
<DIR>   | ..
<DIR>   | d3
<DIR>   | pirana_logs
<DIR>   | temp_dir
run2    | 1-comp iv, alin elim, all | -846.566 | S C | 3.9 |
run2b   | 1-comp iv, alin elim, all | -949.108 | S C | 3.3 |
run2c   | 1-comp iv, alin elim, all | -949.108 | S C | 3.5 | Some notes here...
run40   | 1-comp oral, lin elim, al | -828.267 | SB  | 4.5 |
run40a  | 1-comp oral, lin elim, al |          |     |     |
patab2b |                           |          |     |     |
sdtab2b |                           |          |     |     |

/uma1/users/r/ron/projects/examples/demo/Examples2$
Execu PsN   Nmfe  Refre Outpu Lst   Comme Dupli -Dele Quit
```

Figure 6.1: cPirana interface example

```
perl ~/pirana/cpirana.pl
```

cPirana will use the current folder as it's working folder. To make cPirana more easily accessible, add it as an alias to your `.bashrc` file, e.g.

```
alias cpirana="perl ~/pirana/cpirana.pl"
```

Now you can browse to any folder, and start the program from there using the `cpirana` command.

Chapter 7

Validation

Of course, the information gained from NONMEM or auxiliary tools must be reliable and accurate. Validation of these tools is therefore an important consideration. The main aim of Pirana is to provide overviews of modeling results and providing interfaces to other software (NONMEM, PsN, R, Xpose). As such, Pirana does not perform many calculations of its own. However, Pirana does perform some data parsing, formatting and reporting, i.e. when creating run reports etc.
The Pirana development team has developed tools that perform a validation of parts of this ecosystem. At current, we have two tools available:

psn-validate is a command-line tool that performs a series of pre-specified numerical tests, based on output from PsN tools. In the comparisons, the output from the specific PsN version that is to be validated (*test*-output) is compared to previously obtained output from a PsN version that is trusted, or has been validated in other ways (*reference*-output). The tools is scriptable, and includes R-scripts to perform the numerical comparison. It is however completely flexible, i.e. it allows custom R-scripts to be provided to perform the tests. The tool is available here: `https://github.com/ronkeizer/psn-validate`. Note: recently, the developers of PsN also released their own PsN validation suite with similar functionality.

pirana-validate is a command-line tool that perform a series of numerical test on the parameter estimates outputted by Pirana. The parameters (*test*) are compared to those extracted by PsN's sumo tool (*reference*), for a user-specified library of models and modeling result files. This tool therefore solely focuses on Pirana and PsN's algorithms extraction of parameter estimates from NONMEM output files. Ultimately, this is Pirana's key feature, since Pirana does not perform many calculations, but is primarily a tool to generate overviews of results and allow the modeler to interpret results easier. The tool is currently not publicly available from the Pirana website but can be supplied upon request.

IMPORTANT: We do not claim any responsibility for the outcome or validity of the validation analyeses obtained using these tools. For example, we cannot guarantee that regulatory autharities accept particular validations performed with these tools, nor do we claim the correctness of the validation results. Please align intended validation procedure with the relevant authorities and internal QA/QC procedures.

7.1 Pirana validation tool

While a few models and model outputs are supplied with the Pirana validation tool, it is expected that the user provides a library of models and NONMEM output files (usually `.lst` or `.res` files) as reference. The Pirana validation tool expects a specific directory structure:

```
\val_library
   - \res1
        - \model
   - \res2
   - \res3
```

The valpirana Perl script is invoked from the command line, using e.g.:

```
perl validate.pl -ini=val_pirana20130224.ini
```

`pirana-validate` will then read the ini-file, and loop (alphabetically) through all folders in the folder specified in the ini-file. In each folder, it will extract all parameter estimates and standard error estimates (if available) using Pirana's internal NONMEM output parsing routines. It will then run PsN's `sumo` command, and store the parameters outputted by PsN to memory as well. These will then be compared, allowing for a pre-specified tolerance due to rounding. The ini-file should be specified similar to:

```
[general]
mod=mod
lst=lst
psn_dir=/usr/bin/
pirana_modules=/shared/val/pirana_modules_270
tol=

[lib1]
folder=/shared/val/valpirana_
```

```
[lib2]  # Optionally, specify multiple libraries
folder=
```

Pirana pre-release validation

Before every Pirana release from version 2.4.0 upwards we used (predecessors of) the `pirana-validate` tool to check that the parameter estimates that Pirana extracts from NONMEM output files match those reported by PsN's `sumo` command. For this purpose, we've gathered more than 50 model and results files from multiple modeling groups and many different modelers. While we cannot share the model and output in the validation library, if you would like a report of the validation procedure for a specific version, please contact us.

7.2 PsN validation tool

The PsN validation tool compares output from a *test*-installation of PsN, with output from a trusted (previous) PsN installation. Tests can be implemented for any of the tools in PsN (e.g. `execute`, `bootstrap`, `vpc`, etc), and multiple tests can be performed for each tool (recommended). The tests to be performed are specified in a setup file. Customizable R scripts are used to perform the actual validation step for the tool. Every validation run is started by invoking the valpsn command:

```
./valpsn ex1\_20130201.ini
```

This bash script invokes the main perl script that acutally performs the validation. The script will read in the configuration file, and will perform all the specified validation elements in the sequence specified in the configuration file. The specific tests in the validation run are all implemented in R scripts, and should output `SUCCESS` or `FAILURE`. For every PsN tool a test script is provided with the tool, but the user is encouraged to write custom scripts. More info is available in the specific manual for this tool (available upon request).

Chapter 8

Endnotes

8.1 Acknowledgements

Many Pirana users are recognized for their valuable suggestions and bug-reports, especially those in the Uppsala, Amsterdam, and Cape Town pharmacometrics groups.

8.2 Bug reporting / suggestions

If you find any bugs, please report them at the support section of our website (http://www.pirana-software.com). A ticket will be created, and we will get back to you as soon as possible. Please be as specific as you can about the issue and report version number, system characteristics, and if relevant provide screenshots and model/results files. Requests for further improvement of Pirana are very welcome as well and can also be reported at the support section.