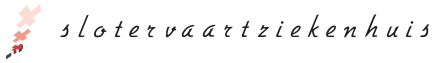
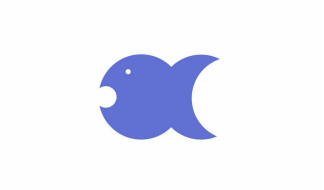
Deptartment of Pharmacy and Pharmacology,
The Netherlands Cancer Institute / Slotervaart Hospital



Piraña

and the Piraña cluster



Version 2.1.0beta
Developer documentation

May 5, 2010

May 5, 2010

Contents

0.1 Outline . . . 3
0.1.1 GUI: Tk toolkit . . . 3
0.1.2 Piraña source code structure . . . 3
0.1.3 Database . . . 4
0.2 Managing and using NONMEM from Piraña . . . 5
0.2.1 NONMEM installations . . . 5
0.2.2 Starting models using NONMEM . . . 5
0.3 Cross-platform portability . . . 6
0.3.1 Tk toolkit . . . 6
0.3.2 Compiling Piraña into a Windows executable . . . 6
0.4 Implementing custom functionality . . . 6
0.4.1 R script functionality . . . 7
0.4.2 Adding new functionality using Perl . . . 7
0.4.3 Perl coding conventions . . . 7

Introduction

Piraña is a graphical user interface (GUI) for performing PK-PD analyses with NONMEM. It provides a platform for model management, editing, running and evaluation. It can be used for modeling locally, but also supports an easy-to-implement distribution infrastructure (PCluster), as well as support for Mosix-clusters.

Piraña is released as open source software, and is still in active development. In this document, the aims of Piraña, the general layout, and several development choices are documented, to help myself and future developers to continue and improve development of Piraña.

Piraña is released under the GNU license. Basically, this means that you can use and distribute this version as much as you want, for free. If you want to help development of future versions of Piraña, either as a programmer or beta-tester, please let me know.

Amsterdam, May 5, 2010
Ron Keizer
ronkeizer@gmail.com

0.1 Outline

Piraña is programmed in Perl and consist of two perl-scripts and several modules. The two perl scripts (pirana.pl and subs.pl), contain most of the functionality for building the GUI. The modules (located in the /pirana_modules folder) contain several specific functionality. For future additions to Piraña, the aim is to include new code as much as possible in separate modules, to ease code-traceability.

Development of Piraña started on the Windows platform, but the aim for future versions is to write and maintain platform-independent code as much as possible. If platform independence is not possible for some functionality, specific code will be added to cope with Linux/Unix/Mac-specific issues.

Module	Description
db.pm	Interface for the SQLite databases
model.pm	Routines for extracting information from NM models and output files
data_inspector.pm	GUI for plotting variables from datasets/NM output files
editor.pm	Built-in code editor with NM-TRAN syntax-highlighting
PsN.pm	Subroutines for interfacing with some PsN functionality
misc.pm	Miscellaneous subroutines

Table 1: Pirana Modules

0.1.1 GUI: Tk toolkit

For implementation of the GUI, Piraña uses the Tk toolkit. Several additional CPAN modules have to be loaded as well. These include: Tk::Balloon, Tk::HList, Tk::ItemStyle, Tk::HdrResizeButton, Tk::Text, Tk::PlotDataset, and Tk::LineGraphDataset, which are available from the CPAN repository. The GUI that is created with Tk looks more or less the same on Windows and Linux/Mac, although slight differences are present, so in some instances OS-specific code has to be added to implemented.

0.1.2 Piraña source code structure

Piraña is initialized in pirana.pl. This creates the main window and frames, and builds the menu. The script also invokes the subroutine *initialize()* (located in internal/subs.pl) which reads the users preferences, software settings, local and network NM-installations, and some other settings from the ini-files (table 2). The ini files specify on each line

the parameter name, the parameter value and a description, separated by commas.

Note: It is my intention to implement reading/writing preferences in a database as well in a future release, but the current implementation suffices for now, and it is also little bit quicker to change settings manually this way.

ini-file	Description
settings.ini	Piraña preferences
software.ini	Links / paths to software like Perl/R/notepad etc.
nm_local.ini	NONMEM installations on local PC
nm_cluster.ini	NONMEM installations on Mosix-cluster (using nmfe)
projects.ini	Locations of project-directories
psn.ini	Default parameters for PsN-commands

Table 2: Pirana Modules

The settings are read by the subroutine read_ini(), which takes the ini-file as input, and returns three hashes for the parameter name, parameter value, description, respectively.

Window size

A compact and a full-screen GUI-layout are provided in the current version of Piraña on Windows. The compact view should not become larger than 1024x600 to allow Piraña to be used on netbooks and smaller screens. The HList widget, which is used for the model/results-overview cannot be stretched automatically to fill the complete window. This implies that when Piraña is maximized in a regular fashion, the main overview widget will stay the same size, and the newly created space is not filled. Therefore, to work around this limitation, Piraña calculates how much new size is created and from there a new width and height for the HList-widget.

0.1.3 Database

For storing notes on models/datasets, results of runs, and keeping logs of executed runs, Piraña uses SQLite-type databases. These databases are contained in the file pirana.dir in each folder that is visited by the user in Piraña. The file is created automatically by Piraña when it finds one or more models in the current folder. The following SQL code is used to create the necessary tables:

```
 CREATE TABLE IF NOT EXISTS model_db model_id
VARCHAR(20), date_mod INTEGER, date_res INTEGER, ref_mod VARCHAR(20),
descr VARCHAR(80), method VARCHAR(12), ofv DOUBLE, suc VARCHAR(2), cov
```

```
VARCHAR(2), bnd VARCHAR(2), sig VARCHAR(4), note TEXT, note_small
VARCHAR(80), note_color VARCHAR(9) )

    ALTER TABLE model_db ADD COLUMN dataset VARCHAR(60)

    CREATE TABLE IF NOT EXISTS table_db ( table_id VARCHAR(50),
date_mod INTEGER, ref_table VARCHAR(50), descr VARCHAR(160), creator
VARCHAR(40), link_to_script VARCHAR(80), note TEXT, table_color
VARCHAR(9) )

    CREATE TABLE IF NOT EXISTS executed_runs ( model_id VARCHAR(20),
descr VARCHAR(80), date_executed INTEGER, name_modeler VARCHAR(30),
nm_version VARCHAR(20), method VARCHAR(12), exec_where VARCHAR(16),
command VARCHAR(120) )
```

In the db.pm module the interface with the database implemented in several subroutines.

0.2 Managing and using NONMEM from Piraña

This section details how Piraña is used as front-end for NONMEM. Basically, there are four main types of running NONMEM locally:

- using nmfe6.bat. This is the standard way of starting NONMEM.
- using NMQual
- using PsN
- using WFN

The first two methods require the user to specify to Piraña where the installation is located. The latter two take care of this themselves: PsN by specifying this in psn.conf and WFN by specifying it in wfn.bat.

0.2.1 NONMEM installations

The locations of the nmfe- and NMQual type are stored in the file ini/nm_local.ini, which are read during initialization of Piraña.

0.2.2 Starting models using NONMEM

0.3 Cross-platform portability

Development of Piraña was first started on Windows in 2006, and initially a Linux version was not considered. However, since using Piraña on a Linux system may have certain advantages (especially when using a Cluster), and the fact that that Perl is a cross-platform language, the cross-platform development of Piraña has become an aim as well. Therefore, as much as possibele, code should be platform-independent. When for specific routines this is not possible (e.g. interacting with the system, executing programs etc.), platform-specific subroutines should be written.

0.3.1 Tk toolkit

In theory, the implementation of Perl/Tk is the same on Windows and Linux (or Mac OSX with the X-window system). However, in practice some differences occur. They include:

- for some widgets it is not possible to specify a border width on Linux
- due to font differences the width or height of some widgets may differ
- on Linux, the background must be specified for each widget, on Windows the background is inherited from the parent widget
- on Windows, icons are not displayed entirely correctly. It is not clear to me why this occurs, likely a bug in Tk

0.3.2 Compiling Piraña into a Windows executable

Since most users will use Piraña on Windows, an executable file is included in the Piraña distribution. This is a compiled version of the Perl script. For the compilation, the Perl Packager (http://search.cpan.org/ãutrijus/PAR-0.85_01/script/pp) is used. To compile Piraña manually from the source files, this command can be used (e.g.):

```
 pp --gui --icon="c:\pirana\images\pirana.ico" -o
pirana.exe pirana.pl --info=ProductName=Pirana
--info=ProductVersion=2.1
```

0.4 Implementing custom functionality

An aim of Piraña is to allow developers to easily add custom functionality to Piraña. Basically, this can be done in two ways:

- Using R scripts. This is the easiest way. Just add them to the scripts folder, and after restarting Piraña the scripts are available from the menu window. Model info is specified to the script, so this can be used to generate run records etc.
- Implementing new perl code. When R scripts don't suffice, or functionality is desired as separate functionality from the Piraña menu, new Perl scripts can be implemented.

Both methods are specified in the following section.

0.4.1 R script functionality

0.4.2 Adding new functionality using Perl

0.4.3 Perl coding conventions

In general, the Google code recommendations are adherede to.
Additionally:

- Subroutines are named as clearly as possible, specifying its functionality already in the subroutine name, e.g. 'run$_{c}ommand_{i}n_{c}onsole()'Theuseofglobalvariablesisavoidedifpossible, use"my"todeclarevariables$

- Subroutines are as much as possible implemented in separate modules.
- After declaration of a subroutine a description follows and a specification whether the subroutine is compatible with Windows / Linux, e.g.:

```
 ### Purpose : Run a command and capture the console
output ### Compat : W+L+
```