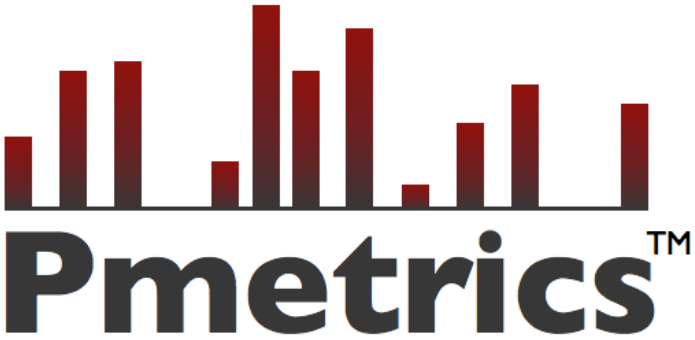


User Manual

April 2013

Package Version 1.0.0

An R package for parametric and non-parametric modeling and simulation of pharmacokinetic and pharmacodynamic systems

Table of Contents

Introduction

Thank you for your interest in Pmetrics! This guide provides instructions and examples to assist users of the Pmetrics R package, by the Laboratory of Applied Pharmacokinetics at the University of Southern California. Please see our website at http://www.lapk.org for more information.

Here are some tips for using this guide.

- Items that are hyperlinked can be selected to jump rapidly to relevant sections.
- At the bottom of every page, the text "User's Guide" can be selected to jump immediately to the table of contents.
- Items in `courier` font correspond to R commands

Citing Pmetrics

Please help us maintain our funding to provide Pmetrics as a free research tool to the pharmacometric community. If you use Pmetrics in a publication, you can cite it as below. In R you can always type `citation("Pmetrics")` to get this same reference.

Neely MN, van Guilder MG, Yamada WM, Schumitzky A, Jelliffe RW. Accurate Detection of Outliers and Subpopulations With Pmetrics, a Nonparametric and Parametric Pharmacometric Modeling and Simulation Package for R. Ther Drug Monit 2012; 34:467–476.

Disclaimer

You, the user, assume all responsibility for acting on the results obtained from Pmetrics. The USC Laboratory of Applied Pharmacokinetics, members and consultants to the Laboratory of Applied Pharmacokinetics, and the University of Southern California and its employees assume no liability whatsoever. Your use of the package constitutes your agreement to this provision.

System Requirements and Installation

There are three required software components which must be installed on your system **in this order**:

1. **R**
2. The **Pmetrics** package for R
3. **gfortran** or some other Fortran compiler

A fourth, highly recommended, but optional component is **Rstudio**, a user-friendly wrapper for R. It can be installed any time after installing R (i.e. step 1 above).

All components have versions for Mac and Windows environments, and 32- and 64- bit processors. All are free of charge.

R

R is a free software environment for statistical computing and graphics, which can be obtained from http://www.R-project.org/. Pmetrics is a library for R.

Pmetrics

If you are reading this manual, then you have likely visited our website at http://www.lapk.org, where you can select the software tab and download our products, including Pmetrics.

Pmetrics is distributed as a package source file archive (.tgz for Mac, .zip for Windows). Do not open the archive. To install Pmetrics from the R console, use the command `install.packages(file.choose())`, and navigate when prompted to the folder in which you placed the Pmetrics package archive (.zip or .tgz) file. Pmetrics will need the following R packages for some functions: chron, Defaults, and R2HTML. However, you do not have to install these if you do not already have them in your R library. They should automatically be downloaded and installed the first time you use a Pmetrics function that requires them, but if something goes awry (such as no internet connection or busy server) you can do this manually.

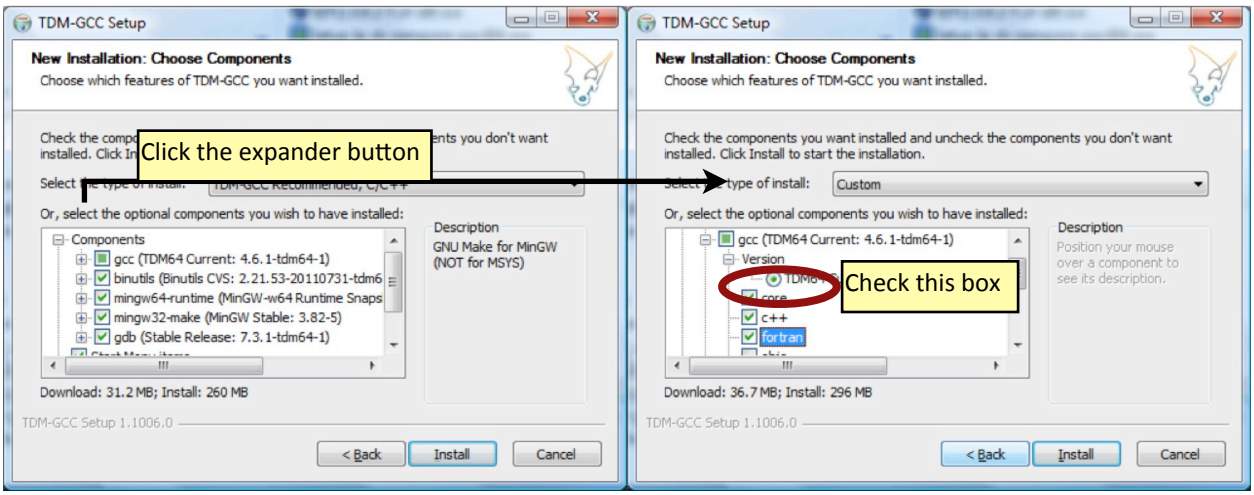
Fortran

In order to run Pmetrics, a Fortran compiler is required. After you have installed Pmetrics, the first time you load Pmetrics into R with the function `library(Pmetrics)`, the program will ask you which Fortran compiler you are using. If you have no compiler, you will have the option to automatically link you to the OS-specific page of our website with explicit instructions and a link to download and install **gfortran** on your system. Details of this procedure follow, but are not relevant if you already have a compiler installed.

For **Mac users** the correct version of gfortran will be downloaded for your system (Mountain Lion 64-bit, Lion 64-bit, Snow Leopard 64- or 32-bit). You will also be provided a link to download and install Apple's Xcode application if you do not already have it on your system. Xcode is required to run gfortran on Macs. As of version 4.3 for Lion, Xcode is available from the App store for free. For Snow Leopard, Xcode is on your installation disk.

NOTE: For Xcode downloaded from the App store (Lion and later), you must additionally install the **Command Line Tools** available in the Xcode Preferences -> Downloads pane.

Windows users need to pay special attention because the the "gcc" installer that provides necessary, common libraries for many programming languages does not by default include gfortran. When gcc is installed, be sure to choose the fortran option to include gfortran, as shown below.



Rstudio

A text editor that can link to R is useful for saving scripts. Both the Windows and Mac versions of R have rudimentary text editors that are stable and reliable. Numerous other free and paid editors can also do the job, and these can be located by searching the internet. We prefer Rstudio.

What This Manual Is Not

We assume that the user has familiarity with population modeling and R, and thus this manual is not a tutorial for basic concepts and techniques in either domain. We have tried to make the R code simple, regular and well documented. A very good free online resource for learning the basics of R can be found at http://www.statmethods.net/index.html. We recognize that initial use of a new software package can be complex, so

please feel free to contact us at any time, preferably through the Pmetrics forum at http:/www.lapk.org or directly by email at contact@lapk.org.

This manual is also not intended to be a theoretical treatise on the algorithms used in IT2B or NPAG. For that the user is directed to our website at www.lapk.org.

Getting Help and Updates

There is an active LAPK forum available from our website at http://www.lapk.org with all kinds of useful tips and help with Pmetrics. Register (separately from your LAPK registration) and feel free to post! Within R, you can also use `help("command")` or `?command` in the R console to see detailed help files for any Pmetrics command. Many commands have examples included in this documentation and you can execute the examples with `example(command)`. Note that, here, quotation marks are unnecessary around `command`. You can also type `PMmanual()` to launch this manual from within Pmetrics as well as a catalogue of all Pmetrics functions. Finally, `PMnews()` will display the Pmetrics changelog.

Pmetrics will check for updates automatically every time you load it with `library(Pmetrics)`. If an update is available, it will provide a brief message to inform you. You can then use `PMupdate()` to update Pmetrics from within R, without having to visit our website. You will be prompted for your LAPK user email address and password. When bugs arise in Pmetrics, you may see a start up message to inform you of the bug and a patch can be installed by the command `PMpatch()` if available. Note that patches must be reinstalled with this command every time you launch Pmetrics, until the bug is corrected in the next version.

As of version 1, Pmetrics has graphical user interface (GUI) capability for many functions. Using `PMcode("function")` will launch the GUI in your default browser. While you are interacting with the GUI, R is "listening" and no other activity is possible. The GUI is designed to generate Pmetrics R code in response to your input in a friendly, intuitive environment. That code can be copied and pasted into your Pmetrics R script. You can also see live plot previews with the GUI. All this is made possible with the 'shiny' package for R.

Currently, the following GUIs are available: `PMcode("NPrun")`, `PMcode("ITrun")`, `PMcode("plot")`. More are coming!

Pmetrics Components

There are three main software programs that Pmetrics controls.

- IT2B is the ITerative 2-stage Bayesian parametric population PK modeling program. It is generally used to estimate parameter ranges to pass to NPAG. It will estimate values for population model parameters under the assumption that the underlying distributions of those values are normal or transformed to normal.
- NPAG is the Non-parametric Adaptive Grid software. It will create a non-parametric population model consisting of discrete support points, each with a set of estimates for all parameters in the model plus an associated probability (weight) of that set of estimates. There can be at most one point for each subject in the study population. There is no need for any assumption about the underlying distribution of model parameter values.
- The simulator is a semi-parametric Monte Carlo simulation software program that can use the output of IT2B or NPAG to build randomly generated response profiles (e.g. time-concentration curves) for a given population model, parameter estimates, and data input. Simulation from a non-parametric joint density model, i.e. NPAG output, is possible, with each point serving as the mean of a multivariate normal distribution, weighted according to the weight of the point. The covariance matrix of the entire set of support points is divided equally among the points for the purposes of simulation.

Pmetrics has groups of R functions named logically to run each of these programs and to extract the output. Again, these are extensively documented within R by using the `help(command)` or `?command` syntax.

- ITrun, ITparse, ITload, ITreport, ERRrun
- NPrun, NPparse, NPload, NPreport
- SIMrun, SIMparse

For IT2B and NPAG, the "run" functions generate batch files, which when executed, launch the software programs to do the analysis. ERRrun is a special implementation of IT2B designed to estimate the assay error polynomial coefficients from the data, when they cannot be calculated from assay validation data (using `makeErrorPoly()`) supplied by the analytical laboratory. The batch files contain all the information necessary to complete a run, tidy the output into a date/time stamped directory with meaningful subdirectories, extract the information, generate a report, and a saved Rdata file of parsed output which can be quickly and easily loaded into R. On Mac (Unix) systems, the batch file will automatically launch in a Terminal window. On Windows systems, the batch file must be launched manually. In both cases, the execution of the program to do the actual model parameter estimation is independent of R, so that the user is free to use R for other purposes.

For the Simulator, the "run" function will execute the program directly within R.

For all programs, the "parse" functions will extract the primary output from the program into meaningful R data objects. For IT2B and NPAG, this is done automatically at the end of a successful run, and the objects are saved in the output subdirectory as IT2Bout.Rdata or NPAGout.Rdata, respectively.

For IT2B and NPAG, the "load" functions can be used to load the above .Rdata files after a successful run. The "report" functions are automatically run at the end of a successful run, and these will generate an HTML page with summaries of the run, as well as the .Rdata files and other objects. The default browser will be automatically launched for viewing of the HTML report page.

Within Pmetrics there are also functions to manipulate data .csv files and process and plot extracted data.

- Manipulate data .csv files: PMreadMatrix, PMcheck, PMfixMatrix, PMwriteMatrix, PMmatrixRelTime, PMwrk2csv
- Process data: makeAUC, makeCov, makeCycle, makeFinal, makeOP, makeNCA, makeErrorPoly
- Plot data: plot.PMcov, plot.PMcycle, plot.PMfinal, plot.PMmatrix, plot.PMop, plot.PMsim, plot.PMdiag, plot.PMpta
- Model selection and diagnostics: PMcompare, plot.PMop (with residual option), PMdiag, PMstep
- Pmetrics function defaults: PMwriteDefaults

Again, all functions have extensive help files and examples which can be examined in R by using the `help(command)` or `?command` syntax.

Customizing Pmetrics Functions

When Pmetrics is loaded with a `library(Pmetrics)` command, it will also load the Defaults package. If not present, it will automatically download it from CRAN. This package allows you to change the default for any Pmetrics function argument. See `?Defaults` for more help, but some key functions are summarized here.

`setDefaults(.name,...)` Where .name is something like `PMreadMatrix`, and `...` is a list of arguments whose default you wish to change. So if you want `PMreadMatrix` to read semicolon delimited files by default instead of comma separated files, use `setDefaults(PMreadMatrix, delim=";")`.

`getDefaults()` or `getDefaults(.name)` The first command will list all functions that have alternative defaults, and the second will list the alternatives for a given function.

`unsetDefaults(.name)` Restores the defaults to normal for a given function.

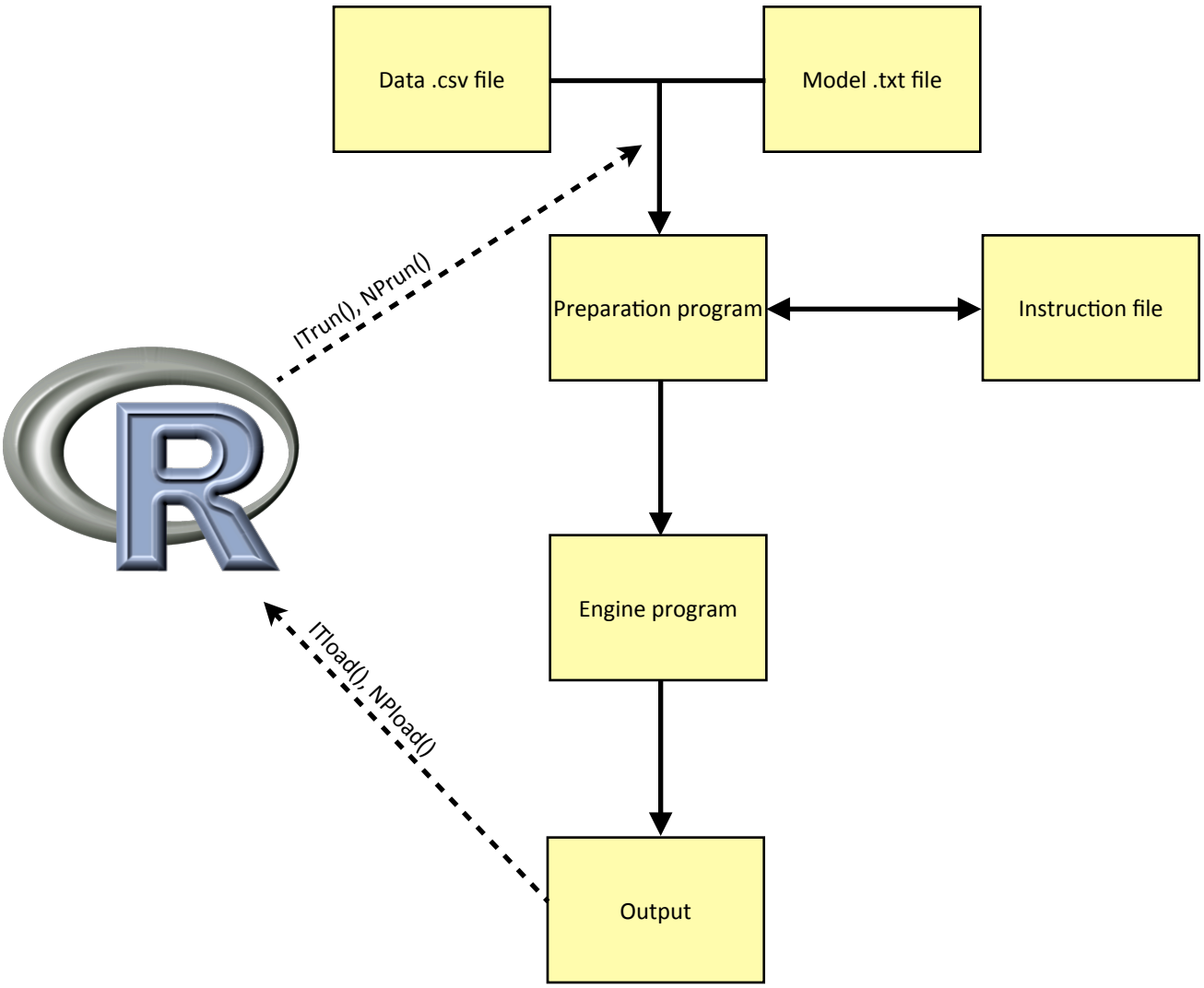
The above functions will manipulate defaults for a single session in R. They are all included in the Defaults package. If you want to make the defaults durable from session to session, use the following function in Pmetrics.

`PMwriteDefaults()` This will save your defaults and they will be restored every time you load Pmetrics.

You supply these files; Pmetrics does the rest!

General Workflow

The general Pmetrics workflow for IT2B and NPAG is shown in the following diagram.



R is used to specify the working directory containing the data .csv and model .txt files. Through the batch file generated by R, the preparation program is compiled and executed. The instruction file is generated automatically by the contents of the data and model files, and by arguments to the NPrun(), ITrun() or ERRrun() commands. The batch file will then compile and execute the engine file according to the instructions, which will generate several output files upon completion. Finally, the batch file will call the R script to generate the summary report and several data objects, including the IT2Bout.Rdata or NPAGout.Rdata files which can be loaded into R subsequently using ITload() or NPload().

Both input files (data, model) are text files which can be edited directly.

Pmetrics Input Files

Data .csv Files

Pmetrics accepts input as a spreadsheet "matrix" format. It is designed for input of multiple records in a concise way. **Please keep the number of characters in the file name ≤ 8.**

Files are in comma-separated-values (.csv) format. Examples of programs that can save .csv files are any text editor (e.g. TextEdit on Mac, Notepad on Windows) or spreadsheet program (e.g. Excel). Click on hyperlinked items to see an explanation.

IMPORTANT: The order, capitalization and names of the header and the first 12 columns are fixed. All entries must be numeric, with the exception of ID and "." for non-required placeholder entries.

POPDATA DEC 11

#ID	EVID	TIME	DUR	DOSE	ADDL	II	INPUT	OUT	OUTEQ	C0	C1	C2	C3	COV...
GH	1	0	0	400	.	.	1	.	.	.	.	.	.	
GH	0	0.5	.	.	.	.	.	0.42	1	0.01	0.1	0	0	
GH	0	1	.	.	.	.	.	0.46	1	0.01	0.1	0	0	
GH	0	2	.	.	.	.	.	2.47	1	0.01	0.1	0	0	
GH	4	0	0	150	.	.	1	.	.	.	.	.	.	
GH	1	3.5	0.5	150	.	.	1	.	.	0.01	0.1	0	0	
GH	0	5.12	.	.	.	.	.	0.55	1	0.01	0.1	0	0	
GH	0	24	.	.	.	.	.	0.52	1	0.01	0.1	0	0	
1423	1	0	1	400	-1	12	1	.	.	.	.	.	.	
1423	1	0.1	0	100	.	.	2	.	.	.	.	.	.	
1423	0	1	.	.	.	.	.	-99	1	0.01	0.1	0	0	
1423	0	2	.	.	.	.	.	0.38	1	0.01	0.1	0	0	
1423	0	2	.	.	.	.	.	1.6	2	0.05	0.2	-0.11	0.002	

POPDATA DEC_11 This is the header for the file and must be in the first line. It identifies the version.

#ID This field must be preceded by the "#" symbol to confirm that this is the header row. It can be numeric or character and identifies each individual. All rows must contain an ID, and all records from one individual must be contiguous. Any subsequent row that begins with "#" will be ignored, which is helpful if you want to exclude data from the analysis, but preserve the integrity of the original dataset, or to add comment lines. IDs should be 11 characters or less but may be any alphanumeric combination. **There can be at most 800 subjects per run.**

EVID This is the event ID field. It can be 0, 1, or 4. Every row must have an entry.

0 = observation

1 = input (e.g. dose)

2, 3 are currently unused

4 = reset, where all compartment values are set to 0 and the time counter is reset to 0. This is useful when an individual has multiple sampling episodes that are widely spaced in time with no new information gathered. This is a dose event, so dose information needs to be complete.

TIME This is the elapsed time in decimal hours since the first event. It is not currently clock time (e.g. 21:30), although this is planned. Every row must have an entry, and within a given ID, rows must be sorted chronologically, earliest to latest.

DUR This is the duration of an infusion in hours. If EVID=1, there must be an entry, otherwise it is ignored. For a bolus (i.e. an oral dose), set the value equal to 0.

DOSE	This is the dose amount. If EVID=1, there must be an entry, otherwise it is ignored.
ADDL	This specifies the number of additional doses to give at interval II. It may be missing for dose events (EVID=1 or 4), in which case it is assumed to be 0. It is ignored for observation (EVID=0) events. Be sure to adjust the time entry for the subsequent row, if necessary, to account for the extra doses. If set to -1, the dose is assumed to be given under steady-state conditions. ADDL=-1 can only be used for the first dose event for a given subject, or an EVID=4 event, as you cannot suddenly be at steady state in the middle of dosing record, unless all compartments/times are reset to 0 (as for an EVID=4 event).
II	This is the interdose interval and is only relevant if ADDL is not equal to 0, in which case it cannot be missing. If ADDL=0 or is missing, II is ignored.
INPUT	This defines which input (i.e. drug) the DOSE corresponds to. Inputs are defined in the model file.
OUT	This is the observation, or output value. If EVID=0, there must be an entry; if missing, this must be coded as -99. It will be ignored for any other EVID and therefore can be ".". There can be at most 150 observations for a given subject.
OUTEQ	This is the output equation number that corresponds to the OUT value. Output equations are defined in the model file.
C0, C1, C2, C3	These are the coefficients for the assay error polynomial for that observation. Each subject may have up to one set of coefficients per output equation. If more than one set is detected for a given subject and output equation, the last set will be used. If there are no available coefficients, these cells may be left blank or filled with "." as a placeholder.
COV...	Any column after the assay error coefficients is assumed to be a covariate, one column per covariate.

Model Files

Model files for Pmetrics are ultimately Fortran text files with a header version of TSMULT... As of Pmetrics version 0.30, we have adopted a very simple user format that Pmetrics will use to generate the Fortran code automatically for you. Version 0.4 additionally eliminates the previously separate instruction file. A model library is available on our website at http://www.lapk.org/pmetrics.php.

Naming your model files. The default model file name is "model.txt," but you can call them whatever you wish. However, **please keep the number of characters in the model file name ≤ 8**. When you use a model file in `NPrun(), ITrun, ERRrun(), or SIMrun()`, Pmetrics will make a Fortran model file of the same name, temporarily renaming your file. At the end of the run, your original model file will be in the /inputs subfolder of the run folder, and the generated Fortran model file will be called "model.for" and moved to the /etc subfolder of the run folder. If your model is called "mymodel.txt", then the Fortran file will be "mymodel.for".

You can still use appropriate Fortran model files directly, but we suggest you keep the .for extension for all Fortran files to avoid confusion with the new format. If you use a .for file as your model, you will have to specify its name explicitly in the `NPrun(), ITrun, ERRrun(), or SIMrun()` command, since the default model name again is "model.txt." If you use a .for file directly, it will be in the /inputs subfolder of the run folder, not in /etc, since you did not use the simpler template as your model file.

Structure of model files. The new model file is a text file with 11 blocks, each marked by "#" followed by a header tag.

#PRImary variables

#COVariates

#SECcondary variables

#BOLus inputs

#INItial conditions

#F (bioavailability)

#LAG time

#DIFferential equations

#OUTputs

#ERRor

#EXTra

For each header, only the capital letters are required for recognition by Pmetrics. The blocks can be in any order, and header names are case-insensitive (i.e. the capitalization here is just to show which letters are required). Fortran is also case-insensitive, so in variable names and expressions case is ignored. Details of each block are next, followed by a complete example.

Primary variables

Primary variables are the model parameters that are to be estimated by Pmetrics or are designated as fixed parameters with user specified values. It should be a list of variable names, one name to a line. Variable names should be 11 characters or fewer. Some variable names are reserved for use by Pmetrics and cannot be used as primary variable names. **The number of primary variables must be between 2 and 32, with at most 30 random or 20 fixed.** On each row, following the variable name, include the range for the parameter that defines the search space. For NPAG, this is absolute, i.e. the algorithm will not search outside this range. For IT2B, the range is a starting range. The simulator will ignore the ranges.

Example:

```
#Pri
KE, 0, 5
V, 0.01, 100
KA, 0, 5
KCP, 0, 5
KPC, 0 , 5
Tlag1, 0, 2
IC3, 0, 10000
FA1, 0, 1
```

Covariates

Covariates are subject specific data, such as body weight, contained in the data .csv file. The covariate names, which are the column names in the data file, can be included here for use in secondary variable equations. The order should be the same as in the data file and although the names do not have to be the same, we strongly encourage you to make them the same to avoid confusion.

Covariates are applied at each dose event. The first dose event for each subject must have a value for every covariate in the data file. By default, missing covariate values for subsequent dose events are linearly interpolated between existing values, or carried forward if the first value is the only non-missing entry. To suppress interpolation and carry forward the previous value in a piece-wise constant fashion, include an exclamation point (!) in any declaration line.

Note that any covariate relationship to any parameter may be described as the user wishes by mathematical equations and Fortran code, allowing for exploration of complex, non-linear, time-dependent, and/or conditional relationships.

Example:

```
#Cov
wt
cyp
IC(!)
```

where IC will be piece-wise constant and the other two will be linearly interpolated for missing values.

Secondary variables

Secondary variables are those that are defined by equations that are combinations of primary, covariates, and other secondary variables. If using other secondary variables, define them first within this block. Equation syntax must be Fortran. It is permissible to have conditional statements, but because expressions in this block are translated into variable declarations in Fortran, expressions other than of the form "X = function(Y)" must be prefixed by a "+" and contain only variables which have been previously defined in the Primary, Covariate, or Secondary blocks.

Example:

```
#Sec
CL = Ke * V * wt**0.75
+IF(cyp .GT. 1) CL = CL * cyp
```

Bolus inputs

By default, inputs with DUR (duration) of 0 in the data .csv file are "delivered" instantaneously to the model compartment equal to the input number, i.e. input 1 goes to compartment 1, input 2 goes to compartment 2, etc. This can be overridden with NBOLUS(input number) = compartment number.

Example:

```
#Bol
NBCOMP(1) = 2
```

Initial conditions

By default, all model compartments have zero amounts at time 0. This can be changed by specifying the compartment amount as X(.) = expression, where "." is the compartment number. Primary and secondary variables and covariates may be used in the expression, as can conditional statemtents in Fortran code. A "+" prefix is not necessary in this block for any statement, although if present, will be ignored.

Example:

```
#Ini
X(2) = IC*V (i.e. IC is a covariate with the measured trough concentration prior to an observed dose)
X(3) = IC3 (i.e. IC3 is a fitted amount in this unobserved compartment)
```

In the first case, the initial condition for compartment 2 becomes the value of the IC covariate (defined in #Covariate block) multiplied by the current estimate of V during each iteration. This is useful when a subject has been taking a drug as an outpatient, and comes in to the lab for PK sampling, with measurement of a concentration immediately prior to a witnessed dose, which is in turn followed by more sampling. In this case, IC or any other covariate can be set to the initial measured concentration, and if V is the volume of compartment 2, the initial condition (amount) in compartment 2 will now be set to the measured concentration of drug multiplied by the estimated volume for each iteration until convergence.

In the second case, the initial condition for compartment 3 becomes another variable, IC3 defined in the #Primary block, to fit in the model, given the observed data.

F (bioavailability)

Specify the bioavailability term, if present. Use the form FA(.) = expression, where "." is the input number. Primary and secondary variables and covariates may be used in the expression, as can conditional statemtents in Fortran code. A "+" prefix is not necessary in this block for any statement, although if present, will be ignored.

Example:

```
#F
FA(1) = FA1
```

Lag time

Specify the lag term, if present, which is the delay after an absorbed dose before observed concentrations. Use the form TLAG(.) = expression, where "." is the input number. Primary and secondary variables and covariates may be used in the expression, as can conditional statemtents in Fortran code. A "+" prefix is not necessary in this block for any statement, although if present, will be ignored.

Example:

```
#Lag
TLAG(1) = Tlag1
```

Differential equations

Specify a model in terms of ordinary differential equations, in Fortran format. XP(.) is the notation for dX(.)/dt, where "." is the compartment number. X(.) is the amount in the compartment. **There can be a maximum of 20 such equations.**

Example:

```
#Dif
XP(1) = -KA*X(1)
XP(2) = RATEIV(1) + KA*X(1) - (KE+KCP)*X(2) + KPC*X(3)
XP(3) = KCP*X(2) - KPC*X(3)
```

RATEIV(1) is the notation to indicate an infusion of input 1 (typically drug 1). The duration of the infusion and total dose is defined in the data .csv file. **Up to 7 inputs are currently allowed.** These can be used in the model file as RATEIV(1), RATEIV(2), etc. The compartments for receiving the inputs of oral (bolus) doses are defined in the #Bolus block.

Outputs

Output equations, in Fortran format. Outputs are of the form Y(.) = expression, where "." is the output equation number. Primary and secondary variables and covariates may be used in the expression, as can conditional statemtents in Fortran code. A "+" prefix is not necessary in this block for any statement, although if present, will be ignored. **There can be a maximum of 6 outputs.** They are referred to as Y(1), Y(2), etc.

Example:

```
#Out
Y(1) = X(2)/V
```

Error

This block contains all the information Pmetrics requires for the structure of the error model. In Pmetrics, each observation is weighted by $1/error^2$. There are two choices for the error term:

1. error = SD * gamma
2. error = $(SD^2 + lamda^2)^{0.5}$ (Note that lambda is only available in NPAG currently).

where SD is the standard deviation (SD) of each observation [obs], and gamma and lambda are terms to capture extra process noise related to the observation, including mis-specified dosing and observation times.

SD is modeled by a polynomial equation with up to four terms: $C_0 + C_1*[obs] + C_2*[obs]^2 + C_3*[obs]^3$. The values for the coefficients should ideally come from the analytic lab in the form of inter-run standard deviations or

coefficients of variation at standard concentrations. You can use the Pmetrics function `PMerrorPoly()` to choose the best set of coefficients that fit the data from the laboratory. Alternatively, if you have no information about the assay, you can use the Pmetrics function `ERRrun()` to estimate the coefficients from the data. Finally, you can use a generic set of coefficients. We recommend that as a start, C_0 be set to half of the lowest concentration in the dataset and C_1 be set to 0.15. C_2 and C_3 can be 0.

In the multiplicative model, gamma is a scalar on SD. In general, well-designed and executed studies will have data with gamma values approaching 1. Poor quality, noisy data will result in gammas of 5 or more. Lambda is an additive model to capture process noise, rather than the multiplicative gamma model. We tend to prefer lambda.

To specify the model in this block, the first line needs to be either L=[number] or G=[number] for a lambda or gamma error model. The [number] term is the starting value for lambda or gamma. Good starting values for lambda are 1 times C_0 for good quality data, 3 times C_0 for medium, and 5 or 10 times C_0 for poor quality. Note, that C_0 should generally not be 0, as it represents machine noise (e.g. HPLC or mass spectrometer) that is always present. For gamma, good starting values are 1 for high-quality data, 3 for medium, and 5 or 10 for poor quality. If you include an exclamation point (!) in the declaration, then lambda or gamma will be fixed and not estimated. Note that you can only fix lambda currently to zero.

The next line(s) contain the values for C_0, C_1, C_2, and C_3, separated by commas. There should be one line of coefficients for each output equation. By default Pmetrics will use values for these coefficients found in the data file. If none are present or if the model declaration line contains an exclamation point (!) the values here will be used.

Example 1: estimated lambda, starting at 0.4, one output, use data file coefficients but if missing, use 0.1,0.1,0,0

```
#Err
L=0.4

0.1,0.1,0,0
```

Example 2: fixed gamma of 2, two outputs, use data file coefficients but if missing, use 0.1,0.1,0,0 for the first output, but use 0.3, 0.1, 0, 0 for output 2 regardless of what is in the data file.

```
#Err
G=2!

0.1,0.1,0,0

0.3,0.1,0,0!
```

Extra

This block is for advanced Fortran programmers only. Occasionally, for very complex models, additional Fortran subroutines are required. They can be placed here. The code must specify complete Fortran subroutines which can be called from other blocks with appropriate call functions.

Reserved Names

The following cannot be used as primary, covariate, or secondary variable names. They can be used in equations, however.

Reserved Variable	Function in Pmetrics
ndim	internal
t	time

x	array of compartment amounts
xp	array of first derivative of compartment amounts
rpar	internal
ipar	internal
p	array of primary parameters
r	input rates
b	input boluses
npl	internal
numeqt	output equation number
ndrug	input number
nadd	covariate number
rateiv	intravenous input for inputs when DUR>0 in data files
cv	covariate values array
n	number of compartments
nd	internal
ni	internal
nup	internal
nuic	internal
np	number of primary parameters
nbcomp	bolus compartment array
psym	names of primary parameters
fa	biovailability
tlag	lag time
tin	internal
tout	internal

Complete Example

Here is a complete example of a model file, as of Pmetrics version 0.40 and higher:

```
#Pri
KE, 0, 5
V0, 0.1, 100
KA, 0, 5
Tlag1, 0, 3

#Cov

wt

#Sec
V = V0*wt

#Lag
TLAG(1) = Tlag1

#Out
Y(1) = X(2)/V


#Err

L=0.4

0.1,0.1,0,0
```

Notes:

By omitting a #Diffeq block with ODEs, Pmetrics understands that you are specifying the model to be solved algebraically. In this case, at least KE and V must be in the Primary or Secondary variables. KA, KCP, and KPC are optional and specify absorption, and transfer to and from the central to a peripheral compartment, respectively.

Brief Fortran Tutorial

Much more detailed help is available from http://www.cs.mtu.edu/~shene/COURSES/cs201/NOTES/fortran.html.

Arithmetic Operator	Meaning
+	addition
-	subtraction
*	multiplication
/	division
**	exponentiation

Relational Operator	Alternative Operator	Meaning
<	.LT.	less than
<=	.LE.	less than or equal
>	.GT.	greater than
>=	.GE.	greater than or equal
==	.EQ.	equal
/=	.NE.	not equal

Selective Execution	Example
IF (logical-expression) one-statement	IF (T >= 100) CL = 10

IF (logical-expression) THEN
  statements
END IF
	IF (T >= 100) THEN
	  CL = 10
	  V = 10
	END IF

IF (logical-expression) THEN
  statements-1
ELSE
  statements-2
END IF
	IF (T >= 100) THEN
	  CL = 10
	ELSE
	  CL = CL
	END IF

How to use R and Pmetrics

Setting up a Pmetrics project

When beginning a new modeling project, it is convenient to use the command `PMtree("project name")`. This command will set up a new directory in the current working directory named whatever you have included as the "project name". For example, a directory called "DrugX" will be created by `PMtree("DrugX")`. Beneath this directory, several subdirectories will be also created: ***Rscript***, ***Runs***, ***Sim***, and ***src***. The ***Rscript*** subdirectory will contain a skeleton R script to begin Pmetrics runs in the new project. The ***Runs*** subdirectory should contain all files required for a run (described next) and it will also contain the resulting numerically ordered run directories created after each Pmetrics NPAG or IT2B run. The ***Sim*** subdirectory can contain any files related to simulations, and the ***src*** subdirectory should contain original and manipulated source data files. Of course, you are free to edit this directory tree structure as you please, or make your own entirely.

Getting the required files to run Pmetrics

When you wish to execute a Pmetrics run, you must ensure that appropriate Pmetrics model .txt and data .csv files are in the working directory, i.e. the Runs subdirectory of the project directory. R can be used to help prepare the data .csv file by importing and manipulating spreadsheets (e.g. `read.csv()`). The Pmetrics function `PMcheck()` can be used to check a .csv file or an R dataframe that is to be saved as a Pmetrics data .csv file for errors. It can also check a model file for errors in the context of a datafile, e.g. covariates that do not match. `PMfixMatrix()` attempts to automatically rid data files of errors. The function `PMwriteMatrix()` can be used to write the R data object in the correct format for use by IT2B, NPAG, or the Simulator.

You can also download sample data and scripts from the Pmetrics downloads section of our website, once you sign in with your LAPK user email address and password. Edit prior versions of model files to make new model files.

Using scripts to control Pmetrics

As you will see in the skeleton R script made by `PMtree()` and placed in the Rscript subdirectory, if this is a first-time run, the R commands to run IT2B or NPAG are as follows. Recall that the "#" character is a comment character.

```
library(Pmetrics)
#Run 1 - add your run description here
setwd("working directory")
NPrun() #for NPAG or ITrun() for IT2B
```

The first line will load the Pmetrics library of functions. The second line sets the working directory to the specified path. The third line generates the batch file to run NPAG or IT2B and saves it to the working directory.

NOTE: On Mac systems, the batch file will be automatically launched in a Terminal window. On Windows systems, the batch file must be launched manually by double clicking the *np_run.bat* or *it_run.bat* file in the working directory.

`ITrun()` and `NPrun()` both return the full path of the output directory to the clipboard. By default, runs are placed in folders numbered sequentially, beginning with "1".

Now the output of IT2B or NPAG needs to be loaded into R, so the next command does this.

```
NPload(run_number)
#or ITload(run_number)
```

Details of these commands and what is loaded are described in the R documentation (`?NPload` or `?ITload`) and in the following section. The run_number should be included within the parentheses to be appended to the names

of loaded R objects, allowing for comparison between runs, e.g. `NPload(1)`. Finally, at this point other Pmetrics commands can be added to the script to process the data, such as the following.

```
plot(final.1)
plot(cycle.1)
plot(op.1,type="pop") or plot(op.1$pop1)
plot(op.1) #default is to plot posterior predictions for output 1
plot(op.1,type="pop",resid=T)
```

Of course, the full power of R can be used in scripts to analyze data, but these simple statements serve as examples.

If you do not use the `PMtree()` structure, we suggest that the R script for a particular project be saved into a folder called "Rscript" or some other meaningful name in the working directory. Folders are not be moved by the batch file. Within the script, number runs sequentially and use comments liberally to distinguish runs, as shown below.

```
library(Pmetrics)

#Run 1 - Ka, Kel, V, all subjects
setwd("working directory")
NPrun() #assumes model="model.txt" and data="data.csv"
NPload(1)
...
```

Remember in R that the command `example(function)` will provide examples for the specified function. Most Pmetrics functions have examples.

Pmetrics Data Objects

After a successful IT2B or NPAG run, an R datafile is saved in the output subdirectory of the newly created numerically ordered folder in the working directory. After IT2B, this file is called "IT2Bout.Rdata", and after NPAG it is called "NPAGout.Rdata". As mentioned in the previous section, these data files can be loaded by ensuring that the Runs folder is set as the working directory, and then using the Pmetrics commands `ITload(run_num)` or `NPload(run_num)`.

Both commands load their respective Rdata files into R, making the contained objects available for plotting and other analysis.

Objects loaded by `ITload(run_num)` and `NPload(run_num)`

Objects	Variables	Comments
op (class: PMop, list)	$pop1, $post1, ...	Population and posterior predictions for each output equation, i.e. 1, 2, ...
	$id	Subject identification
	$time	Observation time in relative decimal hours
	$obs	Observation
	$pred	Prediction based on median of population or posterior parameter value distributions

Objects	Variables	Comments
	$block	Dosing block, usually 1 unless data file contains EVID=4 dose reset events, in which case each such reset within a given ID will increment the dosing block by 1 for that ID
	$obsSD	Calculated standard deviation (error) of the observation based on the assay error polynomial
	$d	Difference between pred and obs
	$ds	Squared difference between pred and obs
	$wd	$d, weighted by the $obsSD
	$wds	$ds, weighted by the $obsSD
final (class: PMfinal, list)	$popPoints	(NPAG only) Data.frame of the final cycle joint population density of grid points with column names equal to the name of each random parameter plus $prob for the associated probability of that point
	$popMean	The final cycle mean for each random parameter distribution
	$popSD	The final cycle standard deviation for each random parameter distribution
	$popCV	The final cycle coefficient of variation for each random parameter distribution
	$popVar	The final cycle variance for each random parameter distribution
	$popCov	The final cycle covariance matrix for each random parameter distribution
	$popCor	The final cycle correlation matrix for each random parameter distribution
	$popMedian	The final cycle median for each random parameter distribution
	$gridpts	(NPAG only) The initial number of support points
	$ab	Matrix of boundaries for random parameter values. For NPAG, this is specified by the user prior to the run; for IT2B, it is calculated as a user specified multiple of the SD for the parameter value distribution

Objects	Variables	Comments
cycle (class: PMcycle, list)	$names	Vector of names of the random parameters
	$ll	Matrix of cycle number and -2*Log-likelihood at each cycle
	$gamlam	A matrix of cycle number and gamma or lambda at each cycle (see item #16 under NPAG Runs below for a discussion of gamma and lambda)
	$mean	A matrix of cycle number and the mean of each random parameter at each cycle, normalized to initial mean
	$sd	A matrix of cycle number and the standard deviation of each random parameter at each cycle, normalized to initial standard deviation
	$median	A matrix of cycle number and the median of each random parameter at each cycle, normalized to initial standard deviation
	$aic	A matrix of cycle number and Akaike Information Criterion at each cycle
	$bic	A matrix of cycle number and Bayesian (Schwartz) Information Criterion at each cycle
cov (class: PMcov, data.frame)	$id	Subject identification
	$time	Time for each covariate entry
	covariates...	Covariate values for each subject at each time, extracted from the raw data file
	parameters...	Mean, median, or mode of Bayesian posterior distribution for each random parameter in the model. Mode summaries are available for NPAG output only, and the default is median. Values are recycled for each row within a given subject, with the number of rows driven by the number of covariate entries

Objects	Variables	Comments
pop (class: PMpop, data.frame) post (class: PMpost, data.frame) NPAG only	$id	Subject identification
	$pred1, ...	Population prior (PMpop) or Bayesian posterior (PMpost) predictions for each output equation, based on mean, median, and mode, as specified by the user and with frequency also specified by the user in the run instructions (see NPAG Runs below, items 23 and 24).
	$block	Same as for PMop objects above
NPdata (class: NPAG, list) ITdata (class: IT2B, list		Raw data used to make the above objects. Please use `?NPparse` or `?ITparse` in R for discussion of the data contained in these objects

Since R is an object oriented language, to access the observations in a **PMop** object, for example, use the following syntax: `op$post1$obs`.

Note that you will place an integer within the parentheses of the loading functions, e.g. `NPload(1)`, which will suffix all the above objects with that integer, e.g. op.1, final.1, NPdata.1. This allows several models to be loaded into R simultaneously, each with a unique suffix, and which can be compared with the `PMcompare()` command (see Model Diagnostics below).

Making New Pmetrics Objects

Once you have loaded the raw (**NPdata** or **ITdata**) or processed (**op**, **final**, **cycle**, **pop**, **post**) data objects described above with `NPload(run_num)` or `ITload(run_num)`, should you wish to remake the processed objects with parameters other than the defaults, you can easily do so with the `make` family of commands. For example, the default for **PMop** observed vs. predicted objects is to use the prediction based on the median of the population or posterior distribution. If you wish to use the mean of the distribution, remake the **PMop** object using `makeOP()`. If you wish to see all the cycle information in a PMcycle object, not omitting the first 10% of cycles by default, remake it using `makeCycle()`.

For all of the following commands, the data input is either **NPdata** or **ITdata**, with additional function arguments specific to each command. Accessing the help for each function in R will provide further details on the arguments, defaults and output of each command.

Command	Description	R help
makeAUC	Make a data.frame of class **PMauc** containing subject ID and AUC from a variety of inputs including objects of **PMop**, **PMsim** or a suitable data.frame	?makeAUC
makeCov	Generate a data.frame of class **PMcov** with subject-specific covariates extracted from the data .csv file. This object can be plotted and used to test for covariates which are significantly associated with model parameters.	?makeCov
makeCycle	Create a PMcycle object described in the previous section.	?makeCycle
makeFinal	Create a PMfinal object described in the previous section.	?makeFinal
makeOP	Create a PMop object described in the previous section.	?makeOP
makeNCA	Create a data.frame (class **PMnca**) with the output of a non-compartmental analysis using **PMpost** and **NPAG** data objects as input. The **PMnca** object contains several columns. • **id**: Subject identification • **auc**: Area under the time-observation curve, using the trapezoidal approximation, from time 0 until the second dose, or if only one dose, until the last observation • **aumc**: Area under the first moment curve • k: Slope by least-squares linear regression of the final 6 log-transformed observations vs. time • **auclast**: Area under the curve from the time of the last observation to infinity, calculated as [Final obs]/k • **aumclast**: Area under the first moment curve from the time of the last observation to infinity • **aucinf**: Area under the curve from time 0 to infinity, caluculated as auc + auclast • **aumcinf**: Area under the first moment curve from time 0 to infinity • **mrt**: Mean residence time, calculated as 1/k • **cmax**: Maximum predicted concentration after the first dose • **tmax**: Time to cmax • **cl**: Clearance, calculated as dose/aucinf • **vdss**: Volume of distribution at steady state, calculated as cl*mrt • **thalf**: Half life of elimination, calculated as ln(2)/k • **dose**: First dose amount for each subject	?makeNCA

Command	Description	R help
makeErrorPoly	This function plots first, second, and third order polynomial functions fitted to pairs of observations and associated standard deviations for a given output assay. In this way, the standard deviation associated with any observation may be calculated and used to appropriately weight that observation in the model building process. Observations are weighted by the reciprocal of the variance, or squared standard deviation. Output of the function is a plot of the measured observations and fitted polynomial curves and a list with the first, second, and third order coefficients.	?makeErrorPoly
makePTA	This function performs a Probability of Target Attainment analysis for a set of simulated doses and time-concentration profiles. Targets (e.g. Minimum Inhibitory Concentrations), the type of target attainment (i.e. %time above target, Cmax:target , AUC:target, Cmin:target, or Cx:target, where x is any time point), and the success threshold (e.g. %time > 0.7 or Cmax:target > 10) can all be specified. Output is a list (class **PMpta**) with two objects • **Results**: a 4 dimensional array with dimension size of [number of doses, number of targets, number of simulated profiles, 1] which gives the target attainment (e.g. Cmax/target) for each dose, target, and profile • **Outcome**: For each dose and target, a summary of the target attainment for all the profiles, including mean, standard deviation and proportion above the success threshold **PMpta** objects can be summarized with summary(x) and plotted with plot(x) .	?makePTA

Command	Description	R help
makePop makePost (NPAG only)	These functions create data.frames of class **PMpop** and **PMpost**, respectively. The **PMpop** or **PMpost** object contains several columns. • **id**: Subject identification • **time**: Times for predicted concentrations. These times are not necessarily at observed times, but at a frequency specified in the NPAG run instructions (see NPAG Runs below, items 23 and 24). • **pred1**: Predictions for output 1. For **PMpop** objects, predictions are based on the mean, median, or mode of the population prior distribution. For **PMpost** objects, they are based on the mean, median, or mode of the Bayesian posterior distribution for each subject. • **pred2, pred3, etc**: If additional outputs exist, they will each be columns in the data.frame, just as for pred1.	?makePop ?makePost

NPAG Runs

In the past, users had to run NPAG once before generating and saving an instruction file which could automate subsequent runs. As of version 0.4, the instruction file is generated automatically using information in the data file, the model file, and arguments to `NPrun()`. However, if `NPrun(auto=F)` is specified, then Pmetrics will allow the user to manually answer all the questions below. **Note, the default `(auto=T)` option means that this section does not apply.**

Answers to the following questions can be saved in an instruction file which can be used for future runs if `(auto=F)`. Instruction files are simply text files with specific entries which can be modified directly by advanced users.

Note that IT2B and NPAG instruction files are NOT interchangeable.

1. Are the files in the current directory? The answer should always be 1.
2. Do an analysis or examine results from prior run? Almost always 1.
3. A warning about using the correct/current model format. Press 1 or some other key and then return.
4. Enter the name of the Fortran model file.
5. For each of the parameters in the model file, specify whether it is to be random (estimated) or fixed (not estimated).
6. What are the ordinary differential equation solver tolerances? Accept the default value by choosing 1, unless advanced.
7. NPAG creates a temporary instruction file in case something happens, so that instructions entered to date can be recovered. Accept the default by choosing 1. You can save with a meaningful filename later. If it already exists, you will be asked if you wish to overwrite the file, which is usually the thing to do.

8. If you have previously run IT2B, you can automatically import the suggested parameter ranges. Choose 1 to do this, and 0 to run NPAG without a previous IT2B run.

    8.1. If you choose option 1, you must specify the name of a FROMxxxx file, typically FROM0001 that you can find in an output directory after a successful IT2B run. Note that the program will then assume that you are using the same .wrk files that IT2B made from your data .csv file. It is strongly recommended to override this and specify a data .csv file. At this point you will jump to #10 below and continue on; however, your answers will often be supplied by the instructions contained in the FROMxxxx file, and you need merely confirm them.

9. Are the instructions coming from the keyboard or a file? If this is the first time, choose 0 for keyboard. If you have a previously saved instruction file that you want to use, choose 1. However, typically if you have an instruction file that you want to use, you will specify this in the R script with `NPrun(instr="filename")` for automated analysis.

    9.1. If you selected keyboard input, you will answer the following questions, otherwise you will be prompted for the name of your instruction file and once loaded, you will verify your previously supplied answers to the following questions.

10. What data input format will you use? The standard format is the matrix block .csv file, so the answer should be 1. Working copy files are an older format. The .csv file is actually converted to these .wrk files, one file per subject, prior to an NPAG run. However, some function will be lost in the Pmetrics package by using .wrk input directly without a .csv file, such as the ability to plot raw subject data via the `plot.PMmatrix()` function.

11. Enter the name of your .csv file now.

12. Enter the total number of unique subjects (defined by ID) in the .csv file.

13. How many of the total number do you want to analyze? Enter 1 if you want to analyze all of them, 0 if you want to analyze a subset.

    13.1. If you entered 0, you will then choose 1 to include specific subjects, or 2 to exclude specific subjects.

    13.2. Enter the inclusion or exclusion subject numbers (not IDs) in order, using a combination of numbers, hyphens and commas. For example: 1,3-5,7,10. Press return and then enter 0 to conclude entry.

14. The program will then open the .csv file and read the number of output equations, reporting each subject as it is read. This can take some time if it is a very large dataset.

15. Enter the boundary values for the random parameters in the model in the form "`min, max`" followed by return.

16. Select how you would like to model assay (observation) error for each output equation. You have four choices. In all four, the standard deviation (SD) of the observation [obs] is modeled by a polynomial equation with up to four terms: $C_0 + C_1*[obs] + C_2*[obs]^2 + C_3*[obs]^3$. You will specify the coefficients C_0, C_1, C_2 and C_3. This information should ideally come from the analytic lab in the form of inter-run standard deviations or coefficients of variation at standard concentrations. You can use the Pmetrics function `PMerrorPoly()` to choose the best set of coefficients that fit the data from the laboratory. Alternatively, if you have no information about the assay, you can use the Pmetrics function `ERRrun()` to estimate the coefficients from the data. Finally, you can use a generic set of coefficients. We recommend that as a start, C_0 be set to half of the lowest concentration in the dataset and C_1 be set to 0.15. C_2 and C_3 can be 0.

    16.1. Error model 1: (SD). Choose this option if you have already run IT2B and multiplied your assay error polynomial by gamma (see next option for a description of gamma).

    16.2. Error model 2: (SD * gamma). Gamma is a scalar to capture additional process noise related to the observation, including mis-specified dosing and observation times. In general, well-designed and executed studies will have data with gamma values approaching 1. Poor quality, noisy data will result in gammas of 5 or more. If you choose this option, you then can specify the starting value of gamma. Good values are 1 for high-quality data, 3 for medium, and 5 or 10 for poor quality.

    16.3. Error model 3: $(SD^2 + lamda^2)^{0.5}$. Lamda is an alternative additive model to capture process noise, rather than the multiplicative gamma model. Good starting values for lambda are 1 times C_0 for good quality data, 3 times C_0 for medium, and 5 or 10 times C_0 for poor quality. Note, that C_0 should generally not be 0, as it represents machine noise (e.g. HPLC or mass spectrometer) that is always present.

16.4. Error model 4: SD = gamma. This model is rarely used and is equivalent to specifying a model with C_0 only, i.e. a constant error regardless of concentration.

17. Once you select the assay error model, for each output equation you are then offered four more options on which assay error polynomial coefficients to use. Choices 1 and 2 are the most commonly used.

    17.1. Choice 1: The default. Use coefficients in the subject record (C_0, C_1, C_2, C_3 in the data .csv file) and if missing, use the default values to be entered in the program (item #18 below).

    17.2. Choice 2: Use the default values to be entered in the program for all subjects, regardless of what is in the data .csv file.

    17.3. Choice 3: To multiply data .csv values and default entered values by a fixed gamma and use them.

    17.4. Choice 0: Specify coefficients on a subject by subject basis, either those in the data .csv file already, the default values entered into the program, or other values.

18. For each output equation, after you have selected the option in item #17 you will be prompted to supply the required information, including the general default values for missing or overridden values in the data .csv file.

19. After assay error pattern and estimates are specified for all output equations, enter the salt fraction of the drug, usually 1. Salt fraction is the percentage of administered compound that contains active drug. For example, the mean salt fraction for theophylline is 0.85. This is not the same as bioavailability, which is the fraction of drug absorbed after non-parenteral administration (e.g. oral) compared to intravenous administration.

20. Enter the grid point index. This number corresponds to the number of grid (support) points which will initially fill the model parameter space. The larger the number of random parameters to be estimated, the more points are required. The program will make a suggestion based on the number of random parameters in the model. The more you choose, the slower the run will be, but results may improve. It is reasonable to choose fewer points early in model exploration and increase in later phases or if poor model fits or lack of convergence are noted. The choices are 1 to 7, corresponding respectively to 2129, 5003, 10007, 20011, 40009, and 80021 points. If you choose 7, you then have an additional choice to select one or more multiples of 80021 points.

21. Enter the maximum number of cycles. This can be 0 or greater. If you enter an integer greater than 0, the engine will terminate at convergence or the number of cycles you specify, whichever comes first. Early in model exploration values of 10 to 100 can be useful, with larger values later in model development. In order to facilitate model comparison, however, we recommend using the same cycle limit for all early models, e.g. 100, rather than choosing 10 for one and 100 for another. If you enter 0, this is the way to test the predictive power of a model on an independent data set and a non-uniform prior must be specified (see #30 below). This means that the engine will only calculate the individual Bayesian posteriors for the new subjects, using the population joint density from a previous run as a Bayesian prior.

22. Information about convergence criteria. Answer 1 or some other keystroke plus Return to acknowledge that you have read it.

23. In order to predict concentrations from a non-parametric distribution, you have the option to use the 1) mean; 2) median or 3) mode of the Bayesian posterior distribution for each subject. We typically use the median first, and then the mean in a separate run and compare the differences.

24. Select the time interval to generate predicted concentrations. Additionally there will also be predictions made at the time of each observation. In general, for most models, predictions every 12 minutes provide sufficient granularity. Smaller values can result in very large files for big populations.

25. Select the default MIC to be used for AUC/MIC ratio reporting. This should generally be set to the default of 1 by choosing 1. You can always extract the AUC later and divide it by any MIC you choose.

26. Enter the value in hours that you want for calculations of AUCs from predicted concentration profiles. The default is 24 hours, which you can accept by entering 1, or 0 if you want to specify a different interval.

27. You are now offered the opportunity to check all of your entries for correctness. Choose 1 each time to indicate correct entries, or 0 to change them.

28. The program will cycle through your subject records again to extract all relevant information. This can take some time if the population is large or individual records are long.

29. Specify the nature of each covariate in the data .csv file. Enter 1 if it is to be considered constant between measurements (e.g. gender) or 2 if values should be extrapolated between observations (e.g. creatinine clearance).

30. For the prior density, choose 1 if it is to be uniform. This means that the initial grid points will be evenly distributed with equal probability within the boundaries specified (in #15 above). If you choose 0, you will be prompted for the name of a file that contains the prior density. This will be DEN0001 unless you have changed its name. It will be found in the output directory of a prior NPAG run. The model used to generate the DEN0001 file must be exactly the same as the current model, including parameter boundaries. However, this option is useful to specify a non-uniform density for two reasons. The first is to test the predictive power of a model on a new set of subjects. Do this in combination with setting the number of cycles to 0 (see #21 above). The second use for a non-uniform prior is to continue a previous run. For example if you only set the number of cycles to 50 to get a rough idea of model fit, you may continue where you left off by specifying the DEN0001 file from the 50-cycle run and continuing with as many additional cycles as you specify in item #21. So, in the example, if you specify 100 cycles in #21, the total number of cycles will be 50 + 100 = 150.

31. Enter 1 if you wish to save all the instructions in an instruction file. If you do this, you can specify this instruction file in Pmetrics by using the `NPrun(instr="yourfile")` option, and including "yourfile" in the working directory with the model .txt file and the data .csv file.

32. Some output will print to the terminal window which contains information that you can ignore while running NPAG from Pmetrics. Press 1 followed by return to begin the NPAG analysis.

33. The NPAG run can complete in seconds for small populations with analytic solutions, or days for large populations with complex differential equations. At the end of a successful run, the results will be automatically parsed and saved to the output directory. Your default browser will launch with a summary of the run.

IT2B Runs

In the past, users had to run IT2B once before generating and saving an instruction file which could automate subsequent runs. As of version 0.4, the instruction file is generated automatically using information in the data file, the model file, and arguments to `ITrun()`. However, if `ITrun(auto=F)` is specified, then Pmetrics will allow the user to manually answer all the questions below. **Note, the default `(auto=T)` option means that this section does not apply.**

Answers to the following questions can be saved in an instruction file which can be used for future runs if `(auto=F)`. Instruction files are simply text files with specific entries which can be modified directly by advanced users.

Note that IT2B and NPAG instruction files are NOT interchangeable.

1. Are the files in the current directory? The answer should always be 1.
2. Do an analysis or examine results from prior run? Almost always 1.
3. A warning about using the correct/current model format. Press 1 or some other key and then return.
4. Enter the name of the Fortran model file.
5. For each of the parameters in the model file, specify whether it is to be random (estimated) or fixed (not estimated).
6. What are the ordinary differential equation solver tolerances? Accept the default value by choosing 1, unless advanced.
7. Are the instructions coming from the keyboard or a file? If this is the first time, choose 0 for keyboard. If you have a previously saved instruction file that you want to use, choose 1. However, typically if you have an instruction file that you want to use, you will specify this in the R script with `ITrun(instr="filename")` for automated analysis.

7.1. If you selected keyboard input, you will answer the following questions, otherwise you will be prompted for the name of your instruction file and once loaded, you will verify your previously supplied answers to the following questions.

8. What data input format will you use? The standard format is the matrix block .csv file, so the answer should be 1. Working copy files are an older format. The .csv file is actually converted to these .wrk files, one file per subject, prior to an IT2B run. However, some function will be lost in the Pmetrics package by using .wrk input directly without a .csv file, such as the ability to plot raw subject data via the `plot.PMmatrix()` function.

9. Enter the name of your .csv file now.

10. Enter the total number of unique subjects (defined by ID) in the .csv file.

11. How many of the total number do you want to analyze? Enter 1 if you want to analyze all of them, 0 if you want to analyze a subset.

    11.1. If you entered 0, you will then choose 1 to include specific subjects, or 2 to exclude specific subjects.

    11.2. Enter the inclusion or exclusion subject numbers (not IDs) in order, using a combination of numbers, hyphens and commas. For example: 1,3-5,7,10. Press return and then enter 0 to conclude entry.

12. The program will then open the .csv file and read the number of output equations, reporting each subject as it is read. This can take some time if it is a very large dataset.

13. Enter the initial boundary values for the random parameters in the model in the form "`min, max`" followed by return.

14. The estimated mean for each parameter value distribution during the first iteration will be the median of the range specified in #13. You now have the option to specify the standard deviation for the parameter value distribution, which by default is half of the range in #13. Choose 1 to accept this (the usual answer) or 0 to change it to something else, expressed as a multiple of the range.

15. In IT2B, the standard deviation (SD) of the observation [obs] is modeled by a polynomial equation with up to four terms: $C_0 + C_1*[obs] + C_2*[obs]^2 + C_3*[obs]^3$. You will specify the coefficients C_0, C_1, C_2 and C_3. You can now choose 1 if every subject has the same coefficients or 0 to use a unique set of coefficients for each subject. The first case is the more usual, when all samples from all subjects are analyzed in the same lab. If samples are analyzed in different labs and you have the assay data from each lab then you would enter 0. This information should ideally come from the analytic lab in the form of inter-run standard deviations or coefficients of variation at standard concentrations. You can use the Pmetrics function `PMerrorPoly()` to choose the best set of coefficients that fit the data from the laboratory. Alternatively, if you have no information about the assay, you can use the Pmetrics function `ERRrun()` to estimate the coefficients from the data (see #15.1.3 below). Finally, you can use a generic set of coefficients. We recommend that as a start, C_0 be set to half of the lowest concentration in the dataset and C_1 be set to 0.15. C_2 and C_3 can be 0.

    15.1. If you choose 1 (one set of coefficients for all subjects) you will then be presented with 3 additional choices.

        15.1.1. Choice 1: Gamma is a scalar to capture additional process noise related to the observation, including mis-specified dosing and observation times. In general, well-designed and executed studies will have data with gamma values approaching 1. Poor quality, noisy data will result in gammas of 5 or more. Choose this option if you wish to fix the assay error coefficients to values either in the data .csv file or as specified in item #16, and to fix gamma to 1.

        15.1.2. Choice 2: Choose this option if you wish to fix the assay error coefficients to values either in the data .csv file or as specified in item #16, but to estimate gamma based on the data. This is the usual option.

        15.1.3. Choice 3: Choose this option if you wish to estimate the assay error coefficients based on your data for use in future runs. Although you can access this option by using either `ITrun()` or `ERRrun()` in R, the instruction files that you save and the generated output files will be different. Therefore, we recommend that if you intend to choose this option, use `ERRrun()` in R, which will generate an ASS0001 file that contains the estimates for C_0, C_1, C_2, and C_3. You can then include this file in the working directory (along with a model .txt file and a data .csv file) to do an IT2B run, supplying the file name in #16.1 below.

15.2. If you choose 0 (unique coefficients for each subject), you will be presented with two choices.

15.2.1. Choice 1: Fix gamma to 1. See the discussion above in #15.1.1.

15.2.2. Choice 2: Estimate gamma based on the data.

15.2.3. You now need to specify where to obtain the values for C_0, C_1, C_2, C_3, either from the data .csv file and from the entry in #16 (Choice 1), or on an individual basis during the IT2B run1 (Choice 0).

16. Enter the values for C_0, C_1, C_2, C_3 that will be used for all patients who do not have values associated with them in the data .csv file.

16.1. You have the option of entering a file name that contains the output of a previous estimation generated by choosing 3 in #15.1.3 above. Usually this file will be called ASS0001 and it must be in the working directory.

17. After assay error pattern and estimates are specified for all output equations, enter the salt fraction of the drug, usually 1. Salt fraction is the percentage of administered compound that contains active drug. For example, the mean salt fraction for theophylline is 0.85. This is not the same as bioavailability, which is the fraction of drug absorbed after non-parenteral administration (e.g. oral) compared to intravenous administration.

18. Enter the convergence criterion. When the difference between log-likelihoods of successive iterations is less than or equal to this criterion, IT2B will converge and terminate. The default is 0.001, which is the typical response.

19. Enter the maximum number of cycles. This can be 1 to 41000, and IT2B will terminate at convergence or the number of cycles you specify here, whichever comes first. Early in model exploration values of 10 to 100 can be useful, with larger values such as 1000 later in model development. In order to facilitate model comparison, however, we recommend using the same cycle limit for all early models, e.g. 100, rather than choosing 10 for one and 100 for another.

20. IT2B can pass parameter ranges to NPAG via a FROMxxxx (usually FROM0001) file. The default ranges to be used in NPAG are 5 times the final IT2B cycle standard deviation above and below the final cycle mean. If your parameters are normally distributed, 5 is a typical number. For log-normally distributed parameters, 3 is a better choice.

21. You are now offered the opportunity to check all of your entries for correctness. Choose 1 each time to indicate correct entries, or 0 to change them.

22. Enter 1 if you wish to save all the instructions in an instruction file. If you do this, you can specify this instruction file in Pmetrics by using the `ITrun(instr="filename")` option, and including "filename" in the working directory with the model .txt file and the data .csv file.

23. The program will cycle through your subject records again to extract all relevant information. This can take some time if the population is large or individual records are long.

24. Specify the nature of each covariate in the data .csv file. Enter 1 if it is to be considered constant between measurements (e.g. gender) or 2 if values should be extrapolated between observations (e.g. creatinine clearance).

25. If you chose unique assay error coefficients for each subject in #15 above, you will now specify whether you wish to use coefficients found in the data .csv file (choice 1), the general coefficients specified in #16 above (choice 2), or a different set that you enter manually now (choice 0) in the form C_0, C_1, C_2, C_3.

26. Some output will print to the terminal window which contains information that you can ignore while running IT2B from Pmetrics. Press 1 followed by return to begin the IT2B analysis.

27. The IT2B run can complete in seconds for small populations with analytic solutions, or days for large populations with complex differential equations. At the end of a successful run, the results will be automatically parsed and saved to the output directory. Your default browser will launch with a summary of the run.

Simulator Runs

The simulator is run from within R. No batch file is created or terminal window opened. However, the actual simulator is a Fortran executable compiled and run in an OS shell. It is documented with an example within R. You can access this by using the `help(SIMrun)` or `?SIMrun` commands from R. In order to complete a simulator run you must include a data .csv file and a model file in the working directory. The structure of these files is identical to those used by IT2B and NPAG. The data .csv contains the template dosing and observation history as well as any covariates. Observation values (the OUT column) for EVID=0 events can be any number; they will be replaced with the simulated values. You can have any number of subject records within a data .csv file, each with its own covariates if applicable. Each subject will cause the simulator to run one time, generating as many simulated profiles as you specify from each template subject. This is controlled from the `SIMrun()` command with the `include` and `nsim` arguments. The first specifies which subjects in the data .csv file will serve as templates for simulation. The second specifies how many profiles are to be generated from each `included` subject.

Simulation from a non-parametric prior distribution (from NPAG) can be done in one of two ways. The first is simply to take the mean, standard deviation and covariance matrix of the distribution and perform a standard Monte Carlo simulation. The second way is what we call semi-parametric, and was devised by Goutelle et al.[1] In this method, the non-parametric "support points" in the population model, each a vector of one value for each parameter in the model and the associated probability of that set of parameter values, serve as the mean of one multi-variate normal distribution in a multi-modal, multi-variate joint distribution. The weight of each multi-variate distribution is equal to the probability of the point. The overall population covariance matrix is divided by the number of support points and applied to each distribution for sampling.

Limits may be specified for truncated parameter ranges to avoid extreme or inappropriately negative values. When you load simulator output with `SIMparse()`, it will include values for the total number of simulated profiles needed to generate `nsim` profiles within the specified limits, as well as the means and standard deviations of the simulated parameters to check for simulator accuracy.

Output from the simulator will be controlled by further arguments to `SIMrun()`. If `makecsv` is not missing, a .csv file with the simulated profiles will be created with the name as specified by `makecsv`; otherwise there will be no .csv file created. If `outname` is not missing, the simulated values and parameters will be saved in a .txt file whose name is that specified by `outname`; otherwise the filename will be "simout". In either case, integers 1 to `nsub` will be appended to `outname` or "simout", e.g. "simout1.txt", "simout2.txt".

Output files from the simulator can be read into R using the `SIMparse()` command (see documentation in R). There is a plot method (`plot.PMsim`) for objects created by `SIMparse()`.

Plotting

There are numerous plotting methods included in Pmetrics to generate standardized, but customizable graphical visualizations of Pmetrics data. Taking advantage of the class attribute in R, a single `plot()` command is used to access all of the appropriate plot methods for each Pmetrics object class.

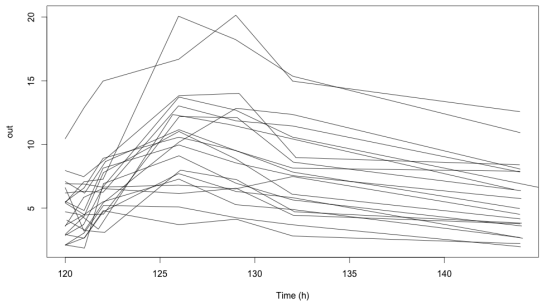
To access the R help for these methods, you must query each method specifically to get details, for `?plot` will only give you the parent function.

Object Classes	Creating functions	R help	Description
PMop	`NPload()`, `ITload()`, `makeOP()`	`?plot.PMop`	Plot population or individual Bayesian posterior predicted data vs. observed. Optionally, you can generate residual plots.

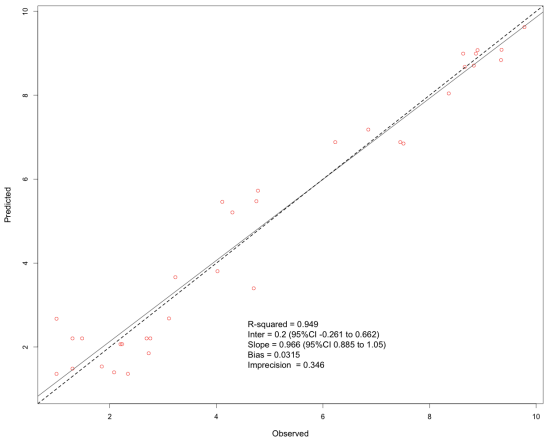
Object Classes	Creating functions	R help	Description
PMfinal	`NPload()`, `ITload()`, `makeFinal()`	`?plot.PMfinal`	Plot marginal final cycle parameter value distributions.
PMcycle	`NPload()`, `ITload()`, `makeCycle()`	`?plot.PMcycle`	Plots a panel with the following windows: -2 times the log-likelihood at each cycle, gamma/lambda at each cycle; Akaike Information Criterion at each cyle and Bayesian (Schwartz) Information Criterion at each cycle, the mean parameter values at each cycle (normalized to starting values); the normalized standard deviation of the population distribution for each parameter at each cycle; and the normalized median parameter values at each cycle. The default is to omit the first 10% of cycles as a burn-in from the plots.
PMcov	`makeCov()`	`?plot.PMcov`	Plots the relationship between any two columns of a PMcov object.
PMmatrix	`PMreadMatrix()`	`?plot.PMmatrix`	Plots raw time-observation data from a data .csv file read by the `PMreadMatrix()` command, with a variety of options, including joining observations with line segments, including doses, overlaying plots for all subjects or separating them, including individual posterior predictions (**post** objects as described above), color coding according to groups and more.
PMsim	`SIMparse()`	`?plot.PMsim`	Plots simulated time-concentration profiles overlaid as individual curves or summarized by customizable quantiles (e.g. 5th, 25th, 50th, 75th and 95th percentiles). Inclusion of observations in a population can be used to return a visual and numerical predictive check.

Object Classes	Creating functions	R help	Description
PMdiag	`PMdiag()`	`?plot.PMdiag`	Plots an npde qqnorm, npde histogram, npde vs. time, npde vs. prediction and standardized visual predictive check to visualize results of simulation based internal model diagnostics accessed with the `PMdiag()` command.
PMpta	`makePTA()`	`?plot.PMpta`	Plots superimposed curves corresponding to each dose, with target (e.g. MIC) on the x-axis and proportion of the simulated time-concentration profiles for the dose with a target statistic (e.g. %time > MIC) above a user-defined success threshold

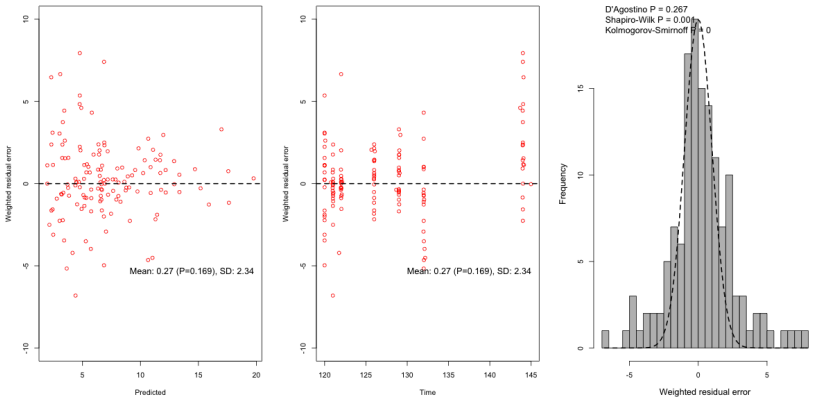
Examples of Pmetrics plots



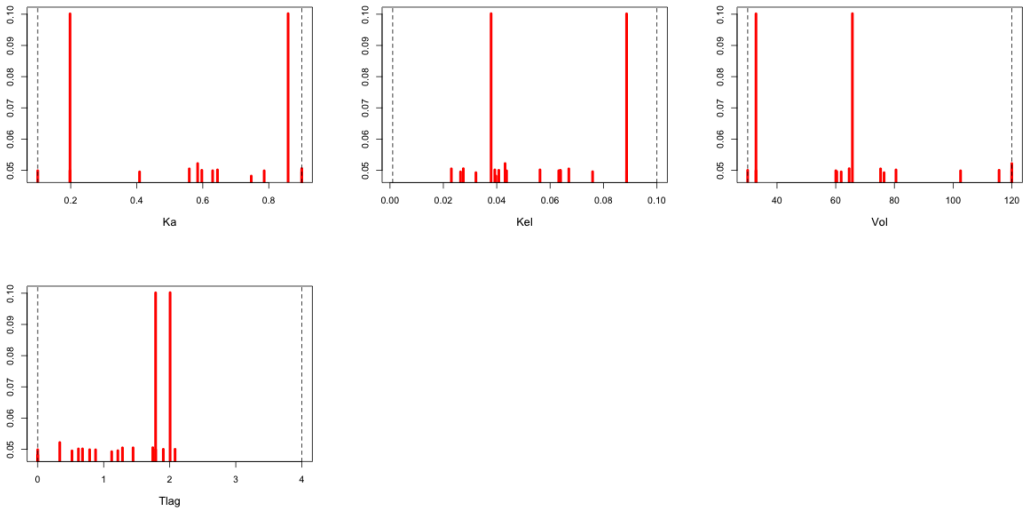
plot(*PMmatrix object*)



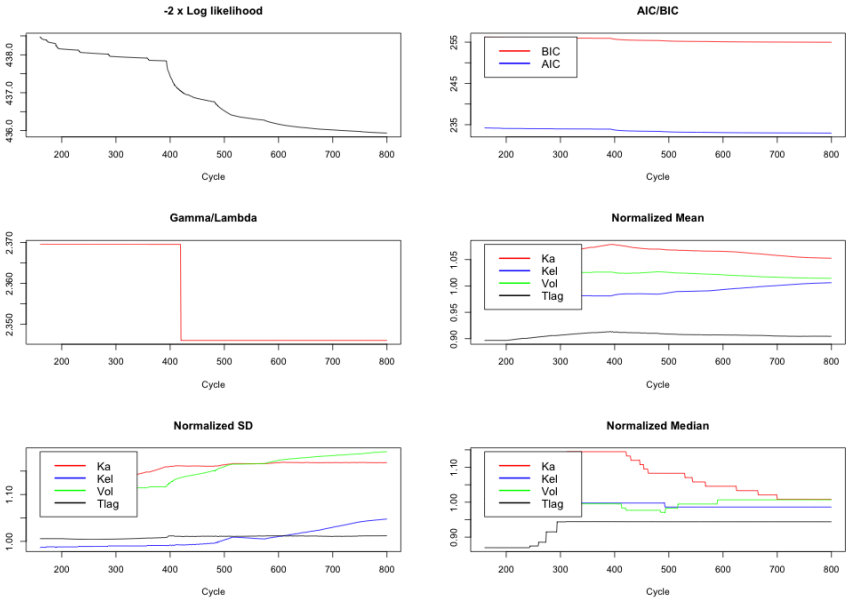
plot(*PMop object*)



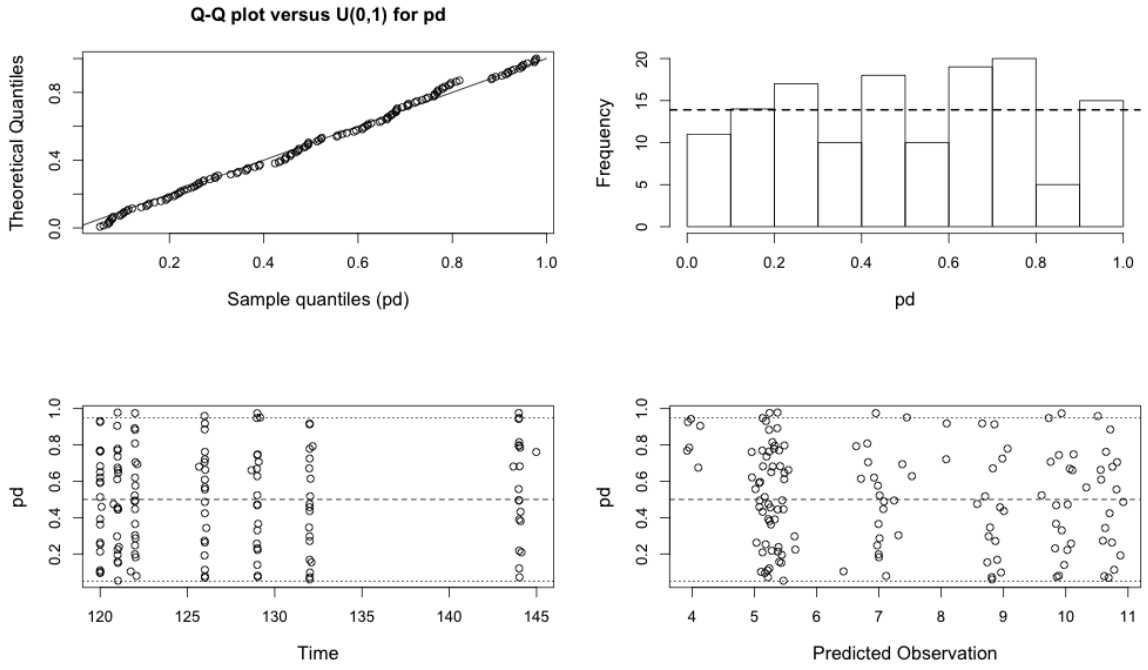
plot(*PMop object, resid=T*)



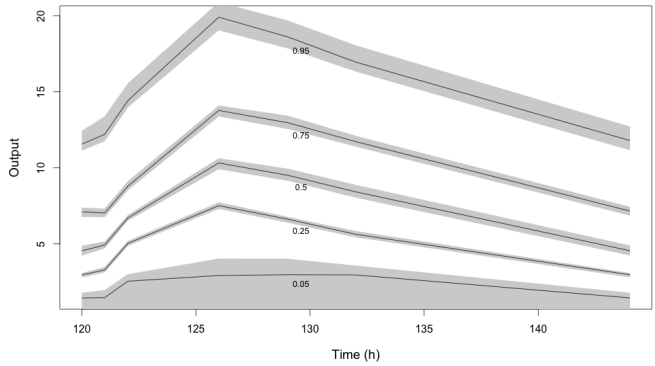
plot(*PMfinal object*)



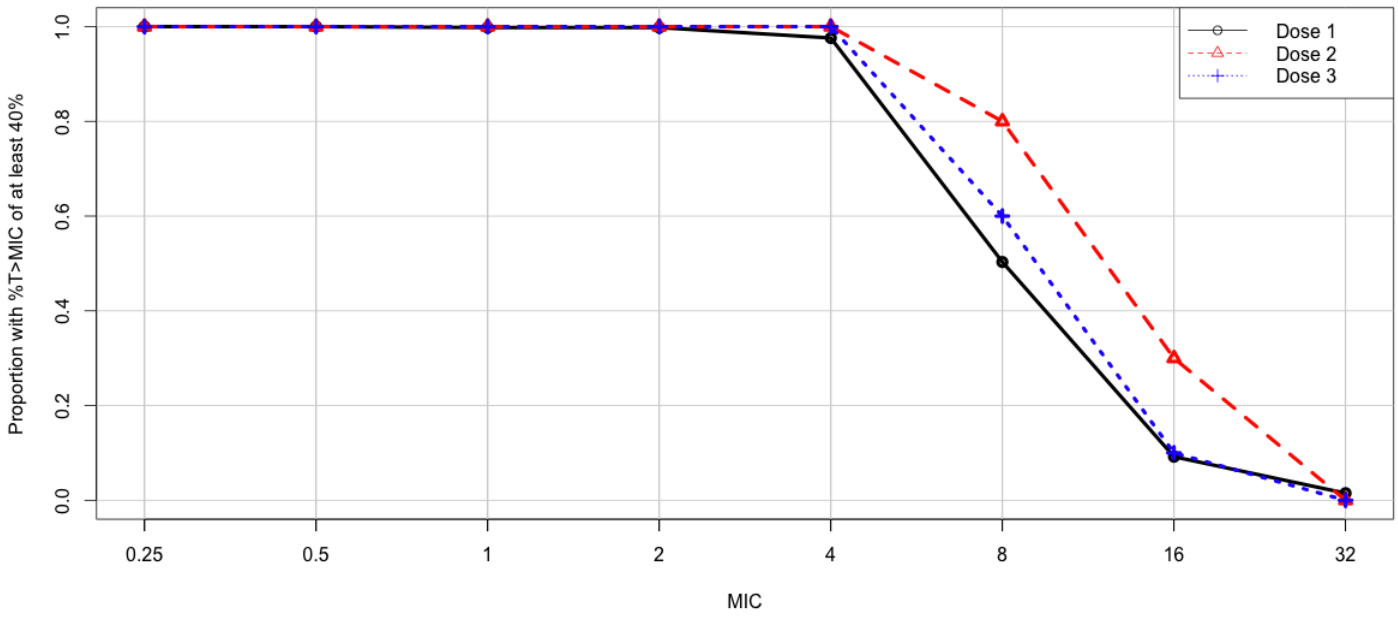
plot(*PMcycle object*)



plot(*PMdiag object*)



plot(*PMsim object*)



plot(*PMpta object*)

1. Model Diagnostics

Internal Validation

Several tools are available in Pmetrics to assist with model selection. The simplest methods are using `PMcompare()` and `plot.PMop()`, via the `plot()` command for a PMop object made by `makeOP()` or by using `NPload()` or `ITload()` after a successful run. `PMstep()` is another option for covariate analysis. All these functions are carefully documented within R and accessible using the `?command` or `help(command)` syntax.

To compare models with `PMcompare()`, simply enter a list of two or more PMetrics data objects. These should be of the NPAG or IT2B class, made either by using `NPload()`/`ITload()` or `NPparse()`/`ITparse()`. Although it is possible to compare models of mixed classes, the validity of this is dubious. The return object will be a data frame with summaries of each model and key metrics such as log-likelihood, final-cycle Akaike Information and Bayesian Information Criteria, and root mean squared errors (RMSE) for observed vs. predictions from the population prior distribution and individual posterior distributions. By specifying the option `plot=T`, observed vs. predicted plots for all the models will be generated. The option to generate residual plots of prediction errors, described next, can be specified with the additional switch `resid=T`, which is ignored if `plot=F`.

As an option to `plot.PMop()`, `resid=T` will generate a residual plot instead of an observed vs. predicted plot. A residual plot consists of three panels: 1) weighted residuals (predicted - observed) vs. time; 2) weighted residuals vs. predictions; 3) a histogram of residuals with a superimposed normal curve if the option `ref=T` is specified (the default). The mean of the weighted residuals (expected to be 0) is reported along with the probability that it is different from 0 by chance. Three tests of normality are reported for the residuals: D'Agostino[2], Shapiro-Wilk, and Kolmogorov-Smirnof. An example is shown in the Plotting section.

`PMstep()` will take a PMcov Pmetrics data object loaded automatically with `NPload()` or `ITload()` after a successful run, and will generate P-values for the relationship of covariates to Bayesian posterior parameter values. Covariates will be tested in a step-wise multivariate linear regression with forwards and backwards elimination, using the `step()` function in the stats package for R, which is a default package. `PMstep()` can help with covariate analysis, although it only performs linear regressions. It is possible to test non-linear relationships

using capabilities of R and the PMcov object, for example with the `nls()` function for non-linear least squares analysis.

Two more complex and time-consuming options are also available: the prediction discrepancy (pd) method of Mentré and Escolano [3], recently recast as a standardized visual predictive check (SVPC) by Wang and Zhang [4]. Both of these can be computed from the same simulation. The basic idea is that each subject in the population serves as a template for a simulation of 1000 further profiles using the population structural model and parameter values joint probability distribution, i.e. together the "population model". The simulated profiles are compared to the observed data, and the pd is generated. The command to generate a PMdiag object is `PMdiag()`, which is documented in R. The same model file and data .csv file used in the NPAG or IT2B run must be in the working directory prior to executing the command. Because of the extensive simulations involved, execution of this command can be slow if the population is large, the model complex, the time horizon long, and/or the number of observations to be simulated per profile is large. There are print and plot methods for PMdiag objects (`print.PMdiag()` and `plot.PMdiag`) both of which are also documented within R. An example of a PMdiag plot is shown in the Plotting section.

Note that simulation from a population model can be a fickle thing, which may lead to errors when trying to execute this command. Parameter value distributions in linear space run the risk of simulating extreme or even inappropriately negative parameter values which can in turn lead to simulated observations far beyond anything corresponding to possible reality. In Pmetrics, the method used to simulate from a prior NPAG (non-parametric) distribution is the split method described above in the Simulator section. Division of the covariance matrix over all the multi-variate distributions in the multi-modal, multi-variate distribution mitigates the problem of extreme values. When using an IT2B (parametric) unimodal, multi-variate distribution, it is likely that extreme values will be simulated. Mitigating techniques include transformation of the model into log-space or switching to an NPAG prior.

There is no command in Pmetrics to automatically generate the simulations necessary for a Visual Predictive Check (VPC), in contrast to the methods described above. VPCs are cumbersome when models include covariates or have heterogeneous dosing/sampling regimens among subjects in the population. It is nonetheless possible to obtain a VPC and numerical predictive check (NPC) using the `plot.PMsim()` command via `plot()` on a PMsim object made with `SIMparse()`. If an observed vs. predicted PMop object made with `makeOP()` is passed to `plot.PMsim()` with the `obspred` argument, the observed values will be overlaid upon simulated profiles if possible, and an NPC will be returned in addition to the plot. The NPC is simply a binomial test for the percentage of observations less than the quantiles specified by the `probs` argument (0.05, 0.25, 0.5, 0.75, 0.95 by default). The simulations for the VPC must be done "manually" using `SIMrun()` and extracted with `SIMparse()` prior to plotting them. It is up to the user to decide if the study population and model is homogeneous enough to justify a VPC.

External Validation

Should you wish to use your population model to test how well it predicts a second population that is separate from that used to build the model (i.e. externally validate your model) you may do that in Pmetrics. After completing an NPAG run, place the same model file (located in the *inputs* subdirectory of the NPAG run whose model you are validating) along with your new (validating) data file in your working directory. So there should be two files in your working directory:

- **model .txt file** This will be the same as for model building NPAG run, found in the /inputs subdirectory.
- **data .csv file** This will be a Pmetrics data input file containing the new subjects for validation.

Next, do the following steps to complete the validation NPAG run.

1. Load the model building run with `NPload(run_num1)` so that its *NPdata* object is in memory.
2. Initiate an NPAG run in Pmetrics as usual, but with an additional argument to specify the model density file which will serve as a non-uniform prior, e.g. `NPrun(model="mymodel.txt", data="validation.csv", prior=NPdata.1, cycles=0)`, where NPdata.1 is and example of

the object loaded in step 1, in this case with `NPload(1)`. Specifying 0 cycles will calculate a Bayesian posterior only for each subject in the validation data set.

3. Complete the NPAG run as usual.
4. Load the results with `NPload(run_num2)` and plot, etc. as usual.

References

1. Goutelle, S. *et al.* Population modeling and Monte Carlo simulation study of the pharmacokinetics and antituberculosis pharmacodynamics of rifampin in lungs. *Antimicrob Agents Chemother* **53,** 2974–2981 (2009).

2. D'Agostino, R. Transformation to Normality of the Null Distribution of G 1. *Biometrika* **57,** 679–681 (1970).

3. Mentré, F. & Escolano, S. Prediction discrepancies for the evaluation of nonlinear mixed-effects models. *J Pharmacokinet Pharmacodyn* **33,** 345–367 (2006).

4. Wang, D. D. & Zhang, S. Standardized visual predictive check versus visual predictive check for model evaluation. *J Clin Pharmacol* **52,** 39–54 (2012).