****

PFIM 3.0

Caroline Bazzoli, Sylvie Retout, Emmanuelle Comets and **France Mentré**

INSERM, U738, Paris, France ; Université Paris 7, Paris, France

Avril 2008

www.pfim.biostat.fr

Read me*

****

PFIM 3.0 is free library of functions.
The University Paris Diderot and INSERM are the co-owners of this library of functions.

Disclaimer

We inform users that the PFIM 3.0 is a tool developed by the Laboratory « Models and methods of the therapeutic evaluation of the chronic diseases "- UMR-S 738, under R and GCC.

PFIM 3.0 is a library of functions. The functions are published after a scientific validation.

However, it may be that only extracts are published.

By using this library of functions, the user accepts all the conditions of use set forth hereinafter.

Licence
1.1.1
This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE UNIVERSITE PARIS DIDEROT OR INSERM OR ITS CONTRIBUTORS BE LIABLE FOR ANY

* The document PFIM3.0_Examples.pdf with fully described examples is also available.

www.pfim.biostat.fr

CONTENTS

1. Description

PFIM 3.0 is a new version of the R function PFIM for evaluation and optimisation of population designs in the context of population pharmacokinetics. It is based on extensions of R functions PFIM1.2 **[1,2,3]** and PFIMOPT1.0 **[4]** for evaluation and optimisation respectively.

This version is extended for multiple response models (1 to K responses) **[5,6]**. It can also be used for single response models instead of PFIM 1.2 or PFIMOPT 1.0 to compute the Fisher information matrix for nonlinear mixed effects models. The development of the matrix **[6]** is the same as for the single response models. This Fisher information matrix is computed for a diagonal variance matrix of the random effects.

The between-subjects variance model may be:

- An additive model : beta + eta
- An exponential model : beta*exp(eta)

where beta is the vector of the fixed effects and eta the vector of between-subjects random effects.

The residual error is additive with a general model for variance for each type of response: $var(\varepsilon)=(\sigma_{inter}+ \sigma_{slope}*f)^2$, where f is the structural model. This variance error model includes the constant variance model ($\sigma_{slope}=0$) or the constant coefficient of variation model ($\sigma_{inter}=0$) as special cases. The parameters σ_{slope} and σ_{inter} are included in the population parameters to be estimated **[3]**.

Conversely to the previous versions of PFIM, there is only one function for both evaluation and optimisation of population pharmacokinetics designs.

In addition to the extension for multiple response models, options have been added for model specification and for optimisation **[7]**, compared to PFIM 1.2 and PFIMOPT 1.0. These options are also available in PFIM Interface 2.1 which however do not handle multiple response models. The model can be written using an analytical form or using a differential equation system. Moreover, with this new version a library of pharmacokinetic models is available.
Concerning optimisation step, an alternative to the Simplex algorithm has been added, the Federov-Wynn algorithm. It allows to optimise design with fixed sampling times in opposite to the Simplex. Thus, the user can have a design adapted to medical constraints.
Options are also available about design optimisation with identical sampling times according to the group and each response. The design can be specified with any sampling times for a response. These specifications are detailed below.

PFIM 3.0 is developed for R 2.4.1 and higher versions.

1.1. Model specification

Models can be specified either with their analytical form or by using system of differential equations.

Moreover, a library of pharmacokinetics model has been added. It has been developed for one or two compartments, for oral (with first order absorption model), bolus or infusion administration and after a single dose, multiple doses or at steady state. Only models with first order

elimination are available in the library of PFIM 3.0. Presently, there are no model with lag time.

1.1.1. Library of models

To use the library of models, the user has to specify the path of the file *librayModels.r* in the model file named by default *model.r* using the following expression:

source(paste(directory.program,dirsep,"libraryModels.r",sep=""))

where the path directory.program is specified during the installation step of PFIM 3.0 presented in the section 2.

An exemple to use the library of models is presented in the section 3.2.

The name of the models in the library of models of PFIM 3.0 are identical to those presented in the library of PFIM Interface 2.1, but the implementation of the model function are different owing to the management of the variable dose in both applications.

The list of the models included in the library is given in Table 1. The table returns every information in order to use the model function chosen.

The model is described by:

- a **name**
- the type of **input**
- the order of **elimination**
- the **number of compartments**
- the parameters used (**parameterisation**)
- the type of **administration** (**sd** : single dose, **md**: multiple dose, **ss**: steady state)
- for each administration type, some variables are required (or not). They are specified in the column named: **Needed variables** (**N:** number of doses, **tau**: interval between two doses, **Tinf**: duration of the infusion)

For models with infusion,the user has to specifiy the duration of infusion (TInf) in the needed variable. The rate of infusion is computed automatically in the function model by the expression: dose/TInf. The variable **dose** has to be specified in the input file (see section 3.3.2).

For example, if one uses after a multiple dose administration, the first order oral absorption with one compartment model (**oral1_1cpt_kaVCl with option md**) from the library, the function of the model used three parameters (**ka**, **Cl** and **V**) and two needed variables (**N**, **tau**): the number of doses (**N**) and the interval between two doses (**tau**).

Table 1. Models included in the library of models

Name	Input	Nb.cpt	Elimination	Parameterisation	Administration	Needed Variable(s)
bolus_1cpt_Vk	IV-bolus	1	1st order	V, k	sd	-
					md	N, tau
					ss	tau
bolus_1cpt_VCl	IV-bolus	1	1st order	V, Cl	sd	-
					md	N, tau
					ss	tau
infusion_1cpt_Vk	IV-infusion	1	1st order	V, k	sd	TInf
					md	TInf, N, tau
					ss	TInf, tau
infusion_1cpt_VCl	IV-infusion	1	1st order	V, Cl	sd	TInf
					md	TInf, N, tau
					ss	TInf, tau
oral1_1cpt_kaVk	1st order	1	1st order	ka, V, k	sd	-
					md	N, tau
					ss	tau
oral1_1cpt_kaVCl	1st order	1	1st order	ka, V, Cl	sd	-
					md	N, tau
					ss	tau
bolus_2cpt_Vkk12k21	IV-bolus	2	1st order	V, k, k12, k21	sd	-
					md	N, tau
					ss	tau
bolus_2cpt_ClV1QV2	IV-bolus	2	1st order	Cl, V1, Q, V2	sd	-
					md	N, tau
					ss	tau
infusion_2cpt_Vkk12k21	IV-infusion	2	1st order	V, k, k12, k21	sd	TInf
					md	TInf, N, tau
					ss	TInf, tau
infusion_2cpt_ClV1QV2	IV-infusion	2	1st order	Cl, V1, Q, V2	sd	TInf
					md	TInf, N, tau
					ss	TInf, tau
oral1_2cpt_kaVkk12k21	1st order	2	1st order	ka, V, k, k12, k21	sd	-
					md	N, tau
					ss	tau
oral1_2cpt_kaClV1QV2	1st order	2	1st order	ka, Cl, V1, Q, V2	sd	-
					md	N, tau
					ss	tau

1.1.2. User defined model

Users can also define their own model analytically or using a system of differential equations.

For user defined models, in PFIM 3.0 the specification of the variable dose can be anywhere in the analytical expression. The name dose should be used and not changed. In the computation of the Fisher Information matrix the dose given in each elementary design will be used.

The following example shows an analytical expression of a one compartment first order absorption model defined by the user. The variable dose in boldface has to be specified in the expression and the value of the dose in the input file (see section 3.3.2).

```
formA<-expression(dose/v*ka/(ka-ke)*(exp(-ke*t)-exp(-ka*t)))
```

The user can give a value to the dose directly in the model, then all elementary design will have the same dose.

The next example describes an analytical expression of a one compartment first order absorption model defined by the user. The dose is specify directly in the expression by the value 100. It is not necessary to specify the dose in the input file in this case.

```
formA<-expression(100/v*ka/(ka-ke)*(exp(-ke*t)-exp(-ka*t)))
```

How to write models in PFIM 3.0 is described precisely and illustrated in the section 3.2.

1.2. Design evaluation

Population designs consist in a set of elementary designs to be performed in groups of subjects, each group being associated with a number or a proportion of subjects. Each elementary design is defined by a number of sampling times to be drawn in each subject.
Design evaluation is based on the computation of the Fisher information matrix. During this process, the expected standard errors on the population parameters with the design are evaluated.

1.3. Design optimisation

PFIM 3.0 allows to optimise exact or a statistical designs. In the case of an exact optimisation, the group structure of the design is fixed: the number of elementary designs, the number of samples per elementary design and the number of subjects per elementary design are given and the design variables to optimise are only the sampling times.
In the case of statistical optimisation, the sampling times (number and allocation) and the proportions of subjects in each elementary design are optimised.

PFIM 3.0 optimises population design using the D-optimal criterion, ie maximising the determinant of the population Fisher information matrix, or, similarly, minimising its inverse.

The Fedorov-Wynn algorithm has been implemented in PFIM 3.0 in addition to the Simplex algorithm. Compared to the Simplex algorithm, the Fedorov-Wynn algorithm better affords high design variables optimisation. Moreover, it considers only pre-specified sampling times, avoiding, clinically unfeasible sampling times. The drawback is the huge number of elementary designs to be created (with corresponding huge number of Fisher information matrices to compute) when the set of allowed sampling times is very large.

1.3.1. Simplex algorithm

The Simplex algorithm optimises statistical or exact designs in constrained intervals, given a total number of samples.
An initial population design needs to be supplied to start the optimisation. The maximum number of elementary designs and the number of sampling times per elementary design are fixed, the sampling times and the proportions of subjects in each elementary design are then optimised. From this initial design, initial vertices for the simplex algorithm are derived, reducing successively each component by 20% (a default value which can be changed) from the original component.
PFIM 3.0 uses the Splus function fun.amoeba from Daniel Heitjan (revised 12/94), which is a translation from the Numerical Recipes for Nelder and Mead Simplex function **[8]**.

1.3.2. Federov-Wynn algorithm

The Fedorov-Wynn algorithm is specifically dedicated to design optimisation problems and has the property to converge toward the D-optimal design **[9-11]**. It optimises statistical designs for a given total number of samples. The sampling times are chosen among a given finite set of times. Minimum and maximum number of samples per subject are specified.
To start the algorithm, an initial population design is then required.
The Fedorov-Wynn algorithm is programmed in a C code and is linked to PFIM 3.0 through a dynamic library, called libfed.dll. Moreover, PFIM 3.0 uses the function combn in the R package “combinat”.

A bug in the previous version of the Fedorov-Wynn algorithm has been corrected. The bug concerns the computation of the number of subjects in the optimised design using the Fedorov-Wynn algorithm.

The only results affected are population designs combining elementary designs with unequal number of sampling times.

eg :

-if the final design is for instance ((1,2,3),(1,3,5)) with frequencies (alpha1, alpha2) it is unaffected by the bug .

-if the final design is for instance ((1,2,3),(1,3)) with frequencies (alpha1, alpha2), it is affected by the bug, and the frequencies will not be correct. In this case the correct frequencies should be respectively (alpha1/3)/(sum) and (alpha2/2)/sum where sum=(alpha1/3)+(alpha2/2) to normalise the sum of the frequencies to 1 ("frequencies" refer to the frequencies reported by the bugged version of the algorithm).

Because of this bug, the final criterion and estimates of SE for the parameters in the model, being computed with the wrong frequencies, is also incorrect, but differences should be small and unlikely to be clinically significant.

The new version of the Fedorov-Wynn outputs the correct frequencies.

1.4. References

[**1].** Retout, Duffull & Mentré. Development and Implementation of the Population Fisher Information Matrix for the Evaluation of Population Pharmacokinetic Designs. *Computer Methods and Programs in Biomedicine.* 2001 65(2): 141-151.

[2]. Retout, Mentré & Bruno. Fisher information matrix for non linear mixed-effects models: evaluation and application for optimal design of enoxaparin population pharmacokinetics. *Statistics in Medicine.* 2002 21: 2623-2639.

[3]. Retout & Mentré. Furthers developments of the Fisher information matrix in nonlinear mixed effects models with evaluation in population pharmacokinetics. *Journal of Biopharmaceutical Statistics.* 2003 13(2): 209-227.

[4]. Retout & Mentré. Optimisation of individual and population designs using Splus. *Journal of Pharmacokinetics and Pharmacodynamics*. 2003 30(6): 417-443.

[5]. Bazzoli, Retout & Mentré. Population design evaluation and optimisation for multiple response models: application to the pharmacokinetics of AZT and AZT-TP. *Congrès de la Société de Pharmacologie Thérapeutique Physiologie.* Toulouse, France, 2007.

[6]. Bazzoli, Retout & Mentré. Population Design in Nonlinear Mixed Effects Multiple Response Models: extension of PFIM and evaluation by simulation with NONMEM and MONOLIX. *Group of Population Optimum Design of Experiments.* Sandwich, England, 2007.

[7]. Retout, Comets, Samson & Mentré. Design in nonlinear mixed effects models: optimisation using the Fedorov-Wynn algorithm and power of the Wald test for binary covariates. *Statistics in Medicine*. 2007 26: 5162-5179.

[8]. Nelder & Mead. A Simplex method for function minimization. *The Computer Journal.* 1964 7: 308-313.

[9]. Fedorov. Theory of Optimal Experiments. Academic Press: New York, 1972.

[10]. Wynn. Results in the construction of D-optimum experimental designs. *Journal of the Royal Statistical Society B.* 1972; 34:133–147.

[11]. Mentré, Mallet & Baccar. Optimal design in random-effects regression models. *Biometrika.* 1997 84: 429–442.

2. Installation

2.1. Pre-requirement

The software R is required. For an optimal use of PFIM 3.0, additional packages might be needed in the R library directory:

- for differential equation system to describe the model: “odesolve” and “nlme” packages
- for the Federov-Wynn algorithm: “combinat” package

The easiest way to install packages is directly from the web. To install the packages odesolve, nlme and combinat start R and choose the Packages item from the menu. Choose Install package(s) from CRAN to install from the web (you will see a list of all available packages pop up -- choose odesolve, nlme and combinat).
To install PFIM 3.0, the user has to download the package named PFIM 3.0 available on the webpage www.pfim.biostat.fr.

2.2. Components

The PFIM 3.0 package includes two main folders called:

- PFIM 3.0
- Examples

The folder PFIM 3.0 is composed of 3 principal files and one folder:

- The 3 principal files are:
  - The main function (program) file (**PFIM3.0.r**)
  - The input file (**Stdin.r**)
  - The model file (**model.r**).

- The folder is called **Program** and contains 10 files of functions:
  - ***Pfim3.0op1.simu.r*** : To compute the Fisher Information matrix to evaluate a population design using an analytical form to describe the model.
  - ***PfimOPT3.0op1.simu.r*** : To compute the Fisher Information matrix to optimise a population design using an analytical form to describe the model
  - ***EQPfim3.0op1.simu.r*** : To compute the Fisher Information matrix to evaluate a population design using a differential equation system to describe the model
  - ***EQPfimOPT3.0op1.simu.r***: To compute the Fisher Information matrix to optimise a population design using a differential equation system to describe the model
  - ***algosimplex.simu.r***: To use the Simplex algorithm
  - ***initfedoR.c*** and ***classfed.h***: To compile the dll
  - ***libFED.dll:*** The dynamic library of the Federov-Wynn algorithm
  - ***algofedorov.simu.r:*** To use the dynamic library libFED.dll
  - ***libraryModels.r:*** To use the library of models

The files in the folder Program should not be changed.

The folder called Examples contains the examples files. The documentation which gives their description is included in the package PFIM3.0 with this documentation.

To install PFIM 3.0, create a directory (for example directory "U:\\My Documents\\PFIM 3.0") and download the package PFIM 3.0.

3. Use

3.1. Working directory

- Create a working directory, for example:

  "U:\\My Documents\\PFIM 3.0_examples\\Example1"

- Copy the files PFIM3.0.r, Stdin.r and model.r in this directory

- In the file "PFIM3.0.r", specify your working directory:

  directory<-"U:\\My Documents\\PFIM 3.0_examples\\Example1"

- Then, specify your program directory i.e. where is the folder called **Program**

  directory.program<-"U:\\My Documents\\PFIM 3.0\\Program"

- Save the file PFIM3.0.r

3.2. Model writing

3.2.1. Analytical form

Write the structural model in the file called "model.r" by default. This model can be given either with an analytical form or as a differential equations system.
The time is denoted by t. In case of analytical form, the model for each response should be written assigned in an object called 'form*i*' where *i* is the letter of the alphabet A,B,C,…. The "form*i*" for all the responses are then grouped in a vector called "form":

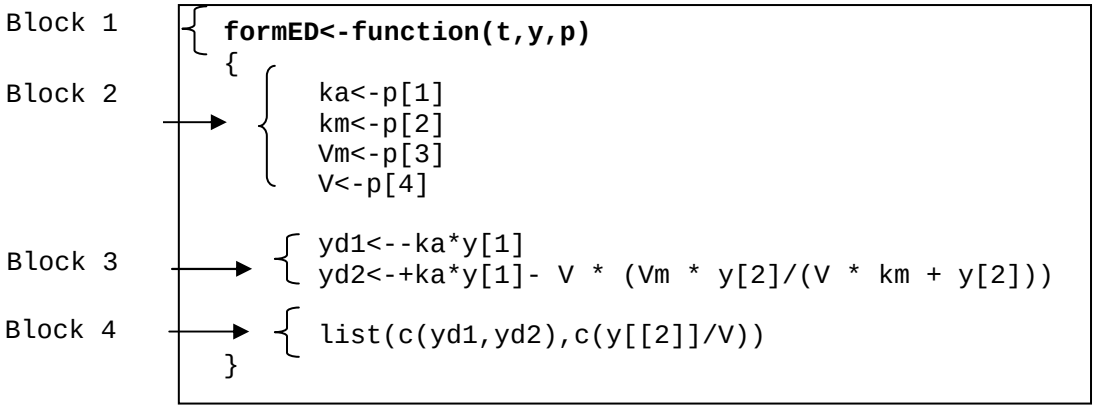
```
form<-c(formA,formB,formC,…)
```

If the model for a form is defined over intervals by different expressions, write these expressions in the object 'form*I*', which in this case is a vector of expressions. For example, if you want to give three expressions for the first response:

```
formA<-c(form1,form2,form3)
```

3.2.2. Differential equation system

For a differential equations system, user must supply a function called "formED":



In the **block 1** specifies function formED with the default arguments of the function:

- 't' is the current time point in the integration
- 'y' is the current estimate of the variables in the ode system
- 'p' is the vector of parameters

In **Block 2**, the user has to be defined the name of the parameters to be estimated. They are attributed to each component of the vector of parameters "p".

Block 3 stands for the definition of the differential equation system, with here, two exchange components and two lines.
Block 4, the last line returns a list of two objects: the first object is the vector of the derivatives of the system; the second object indicates the measures of interest (in this case only one measure is considered the concentration in the second compartment scaled by the volume).
The initial values of the system have to be specified in the input file presented in the section 3.3. Here, the previous figure describes a one compartment model first order absorption and Mickaelis-Menten elimination.

This extension to differential equations system requires the use of the lsoda function included in the library "odesolve" (version 0.5 -12 by R. Woodrow Setzer, 25 October 2004) and of the fdHess function included in the library "nlme" developed by Jose Pinheiro and Douglas Bates.
The lsoda function uses a function of the same name written in Fortran by Linda R. Petzold and Alan C. Hindmarsh. This function solves system of differential equations using the Adams method, a predictor – corrector method for non-stiff systems; it uses the BDF (Backward Differentiation Formula) for stiff systems.
The fdHess is used for numerical derivation. It evaluates an approximate gradient of a scalar function using finite differences.

Several examples are presented below with the different way of writing models.

3.2.3. Examples

3.2.3.1. Example 1: Single response

Analytical form

```
formA<-expression((dose/v * ka)/(ka - ke) * (exp( - ke * t) - exp( - ka * t)))

form<-c(formA)
```

Analytical form with the library of models

In this illustration, the user creates a one response model using the model function implemented in the library (**Oral1_1cpt_kaVCl**) *describing a one compartment oral absorption after a multiple dose administration (**md**). N and tau are the needed variables and thus, they have to be specified by the user in the function model. Here, we have five oral administration doses with an interval between two doses equal to twelve hours.*

```
source(paste(directory.program,dirsep,"libraryModels.r",sep=""))

formA<-Oral1_1cpt_kaVCl_md(N=5,tau=12)[[1]]
form<-c(formA)
```

Differential equations system

In this illustration, the user creates a one response model using a differential equation system describing a one compartment first order absorption model. It is parameterized in ka (constant of absorption), ke (constant of elimination) and v (volume of distribution). The measure of interest is the concentration in the first compartment scaled by v.

```
formED<-function(t,y,p)
{
      ka<-p[1]                          } Block 2
      ke<-p[2]
      v <-p[3]

      yd1<-ka*y[2]-ke*y[1]              } Block 3
      yd2<--ka*y[2]

      list(c(yd1,yd2),c(y[1]/v))        } Block 4
}
```

3.2.3.2. Example 2: Two responses

Analytical form

In this illustration, the user creates a two response model using an analytical form to describe the model. The first response is described by a one compartment model with first order absorption. A second compartment is added with a first order rate constant, corresponding to the second response.

```
      form1<-expression((dose*Ka/(V*(Ka-(Cl/V+R)))*
((exp(-(Cl/V+R)*t)/(1-exp(-(Cl/V+R)*12)))-(exp(-Ka*t)/(1-exp(-Ka*12))))
))

formA<-c(form1)                                    } First response

      form1<-expression((dose*Ka/(1/R)*((exp(-(Cl/V+R)*t)/((Ka-
(Cl/V+R))*((Clm/R)/(1/R)-(Cl/V+R))*(1-exp((Cl/V+R)*12))))+exp(-
Ka*t)/(((Cl/V+R)-Ka)*((Clm/R)/(1/R)-Ka)*(1-exp(-Ka*12)))+exp(-
((Clm/R)/(1/R))*t)/((Ka-(Clm/R)/(1/R))*((Cl/V+R)-(Clm/R)/(1/R))*(1-exp(-
(Clm/R)/(1/R) *12))))))

formB<-c(form1)                                    } Second response

form<-c(formA,formB)
```

Differential equations system

In this illustration, the user creates a two response model using a differential equation system. The first measure of interest is the concentration in the compartment 2 scaled by the volume and the second measure of interest is the concentration in the compartment 3.

```
formED<-function(t,y,p)
{
ka<-p[1]                                           } Block 2
cl<-p[2]
V<-p[3]
clm<-p[4]
R<-p[5]

yd1<--ka*y[1]                                      } Block 3
yd2<-ka*y[1]-cl/V*y[2]-R*y[2]
yd3<-R*y[2]-clm*y[3]

list(c(yd1,yd2,yd3),c(y[2]/V,y[3]))                } Block 4

}
```

3.2.3.3. Example 3: Two responses using a PK model from the library

Analytical form with the library of models

In this illustration, the user creates a two response model using the model function implemented in the library (**Oral1_1cpt_kaVCl**) *describing a one compartment model oral absorption after a single dose administration* (***sd***) *for the first response. No variable is needed. An Imax model with constant baseline characterises the second response.*

```
source(paste(directory.program,dirsep,"libraryModels.r",sep=""))

formA<-Oral1_1cpt_kaVCl()[[1]]
formB<-paste("-Imax*",formA,"/(C50+",formA,")+S0")
```

```
formB<-parse(text=formB)
form<-c(formA,formB)
```

3.3. Population Information input

Conversely to the previous version of PFIM, there is not one function for evaluation and another one for optimisation of the population pharmacokinetics design. Only, one input file has to be filled called by default: stdin.r.
To specify evaluation or optimisation designs you have to complete the "run" object:

- **run**: character string for function selection:
  - "**EVAL**" for evaluation
  - "**OPT**" for optimisation

According to the choice, specific objects notified below must be specified.

3.3.1. General objects required for Evaluation and Optimisation

This section shows what are the common objects required for both evaluation and optimisation of a population design. In the file stdin.r, the following R objects must be created:

- **project**: character string indicating the name of the project
- **file.model**: character string indicating where to find the structural model
- **output**: character string indicating in which file the results should be printed
- **nr**: value indicating the number of responses in the model
- **parameters**: vector of p character strings for the names of the fixed effects parameters
- **beta**: vector of the p fixed effects parameters values
- **omega**: vector of the p variances of the random effects should be given
- **sig.inter*i*** and **sig.slope*i*** : values of the parameters for the residual variance error model given by $var(\varepsilon_i)=(\sigma_{inter_i} \cdot \sigma_{slope_i} * f_i)^2$ for each response *i*, with i = A,B,C,….
- **Trand:** type of between-subject variance (or random effects) model:
  - 1 for additive between-subject variance model
  - 2 for exponential model
- **prot*i***: list of vectors of elementary designs for each response *i* with i = A,B,C,…. Each vector contains the sampling times of the corresponding elementary design for the respective response. The size of the vector for each response has to be the same.
  For example, if there are two responses, the user must specify:
  protA<-…

prot*B*<-…

If there are several responses with several elementary designs, the user can specify by NULL if a group do not have samples for one response. Example:

prot*A*<-list(c(1,3,6,12),c(18,20,24))
prot*B*<- list(c(1,3,6,12),c(NULL))

In the present version, the Federov-Wynn does not work when there is a NULL design for one response.

- **subjects** : vector of the q numbers of subjects for each elementary design. **Only for optimisation**, this object is a vector of the q initial proportions or of the numbers of subjects in each elementary design.

Additional R objects are required to be created for analytical model specification or differential equations system and graphical options.

3.3.1.1. Model option

- **modelform** : character string indicating either a model given under differential equation systems (“**DE**”) or analytical form (“**AF**”)

For analytical model specification only

- **dose.identical** : logical value : ‘T’ if the dose is the same for all elementary designs, ‘N’ if not.

- **dose**: value of the dose if dose.identical==T; if not, vector of the q doses for each elementary design.
If one uses infusion models implemented in the library of models, the dose has to be specified here. The rate of infusion is computed in the function model by the expression: dose/TInf.

- **bound*i***: vector of bounds specific to each response *i* with i=A,B,C,…. If the model is defined over intervals by different expressions, give in that vector, the values of each time interval for the response *i*. The first expression will be use for the first time interval, the second expression for the second time interval …

For differential equations system specification only

- **time.condinit**: initial time at which initial conditions are given.

- **condinit.identical**: logical value: 'T' if the initial conditions are the same whatever the elementary design, 'F'if not.

- **condinit**: initial values of the system at the initial time, given into an expression. If condinit.identical==T, enter once the expression of the initial values of the system at the initial time; else, enter the vectors of the initial conditions for each elementary design. If initial values depend on parameters to be estimated, enter this parameter into the expression without any quotation marks

- **RtolEQ**: relative error tolerance, either a scalar or an array as long as 'y'. See details in help for lsoda function. Default value is 1e-06.

- **AtolEQ**: absolute error tolerance, either a scalar or an array as long as 'y'. See details in help for lsoda function. Default value is 1e-.06

- **Hmax**: an optional value for the maximum integration stepsize. Default value is Inf.

*From help for lsoda: "The input parameters 'RtolEQ', and 'AtolEQ' determine the error control performed by the solver. The solver will control the vector *e* of estimated local errors in *y*, according to an inequality of the form max-norm of (*e*/*ewt*) <= 1, where *ewt* is a vector of positive error*
*weights. The values of 'RtolEQ' and 'AtolEQ' should all be non-negative. The form of *ewt* is:*

RtolEQ* * abs(*y*) + *AtolEQ

where multiplication of two vectors is element-by-element.

If the request for precision exceeds the capabilities of the machine, the Fortran subroutine lsoda will return an error code; under some circumstances, the R function 'lsoda' will attempt a reasonable reduction of precision in order to get an answer. It will write a warning if it does so."

3.3.1.2. Graph option

This list of objects allows to draw a graph with the evaluated design (evaluation step) or the optimised design (optimisation step).
If the user wants a graph, he has to specify it in the following object.

- **graph.logical**: logical value; if T, draws the graph of the predicted output(s) and the sampling times.

If the user has filled in by T the previous object, he has to specify the following objects.

- **names.data:** vector of character string for the names of Y axis for each graph that correspond of each type of measurement (length of this vector must equal to the number of responses).

- **log.logical**: character string for controls logarithmic axes for the graphical representation. Values “xy”, “x” or “y” produce log-log or log-x or log-y axes. Standard graphic is given by log.logical<-F

- **graph.infi** and **graph.supi**: vector of lower and upper sampling times for the graph for each response *i* with i=A,B,C, …. For example for a single response model, representation in the interval [0-60] is specified by **graph.infA<-c(0)** and **graph.supA<-c(60). If any lower and upper sampling times for the graph are specified, by default for each response the lower sampling time is 0 and the upper sampling time is the maximum sampling time of the initial design.**

- **y.range**: vector of lower and upper values for the graphical representation. They are identical of each response. By default, the value is NULL. For example, representation in 1 interval [0-10] is specified by **y.range=c(0,10).**

3.3.2. Objects required only for Optimisation

3.3.2.1. Population design

- **subjects**: **only for optimisation**, this object is a vector of the q initial proportions (if subjects.input=2) or of the numbers of subjects in each elementary design (if subjects.input=1).

- **subjects.input**: **1** if the subjects per elementary designs are given as numbers, **2** if they are given as proportions

- **Ntot:** total number of samples. Ntot is required if the subjects per elementary design are given as proportions, i.e., only if subjects.input=2

3.3.2.2. Algorithm option

The user with PFIM 3.0 can optimise design with identical time in each elementary design for each response. To trigger this option, the user needs to specify it in the next objects:

- **identical.times**: **T** if identical sampling times for each response within an elementary design else **F**

The choice of the algorithm is specify in the following object.

- **algo.option**: character string for algorithm selection. “**SIMP**” for Simplex algorithm selection; “**FW**” for Fedorov-Wynn algorithm selection

For each algorithm, the user need to specify particular objects described below.

For Simplex algorithm specification only

- **subjects.opt**: logical value: 'T' is the optimisation of the proportions or the number of subjects is required; 'F' if not.

- **loweri** and **upperi**: vector of lower and upper admissible sampling times for each response i with i = A,B,C, …. For example if there is a single response model, representation in 2 intervals [0-12], and [48-60] is specified by upperA<-c(0,48) and lowerA<-c(12,60)

- **delta.time**: numeric value for the minimum delay between two successive sampling times

- **iter.print**: logical value to print the iterations (T) or not (F)

- **simplex.parameter**: percent of change from initial design for the initial vertices of the simplex algorithm building (default 20%).

- **Max.iter**: a positive integer specifying the maximum number of iterations allowed (default = 5000)

- **Rctol**: a positive numeric value specifying the tolerance level for the relative convergence criterion of the Simplex algorithm (default = 1e-6)

For Fedorov-Wynn algorithm only

- **nwindi**: numeric value for the number of sampling intervals for each response i with i=A,B,C, ….

- **sampwini**: list of vector of the allowed sampling times for each sampling interval for each response i with i = A,B,C, ….

- **nsampi**: list of vector of allowed numbers of points to be taken from each sampling interval for each response i with i = A,B,C, ….

- **nmaxptsi**: numeric value for the maximum number of sampling times per subject for each response i with i = A,B,C, ….

- **nminptsi**: numeric value for the minimum number of sampling times per subject for each response i with i = A,B,C, ….

For Fedorov-Wynn algorithm use, the initial population design specified in prot must involve elementary designs with number of samples per subject and sampling times in accordance with sampwin, nsamp, nmaxpts and nminpts for each response.

4. Run

Once the input file and the model file are filled in, the user can run PFIM 3.0. Load the main function PFIM() implemented in the file PFIM3.0.r. To do that, choose the File item from the menu. Select “Source R code”; click on the right directories up to the file PFIM3.0.r. The user can also load the file by typing the command in the Command Window:

```
source("U:\\My Documents\\PFIM 3.0_examples\\Example1\\PFIM3.0.r")
```

Call the R function in the Commands window: **PFIM()**

5. Results

The results are written in the output file called stdout.r by default or with the name specified in the input file. This file is different when only evaluation is performed or when optimisation is performed. It is therefore detailed in next sections respectively for evaluation and for optimisation.

5.1. Evaluation output file and objects

Figure 1 represents the output file from the design evaluation described in the Example documentation in the section 1.2.1.

The user can read on the Figure 1:

1 The name of the function used: PFIM 3.0.

2 The name of the project and the date.

3 A summary of the input: model(s), sampling times in the elementary designs for each model(s), doses or initial conditions and subjects corresponding to those designs, residual variance error model for each model(s), residual between-subject variance model, initial population design, errors tolerances for the solver of differential equations system if used. *The figure shows a two responses model(analytical form) with a group described by four sampling times for each response for 50 subjetcs. The dose is equal to 300.*

4 The population Fisher information matrix, a dim*dim symmetric matrix where dim is the total number of population parameters to be estimated.

5 The value of each population parameter with the expected standard error on each parameter and the corresponding coefficient of variation.

6 The value of the determinant of the Fisher information matrix and the value of the criterion (determinant^(1/dim)) where dim is the total number of population parameters.

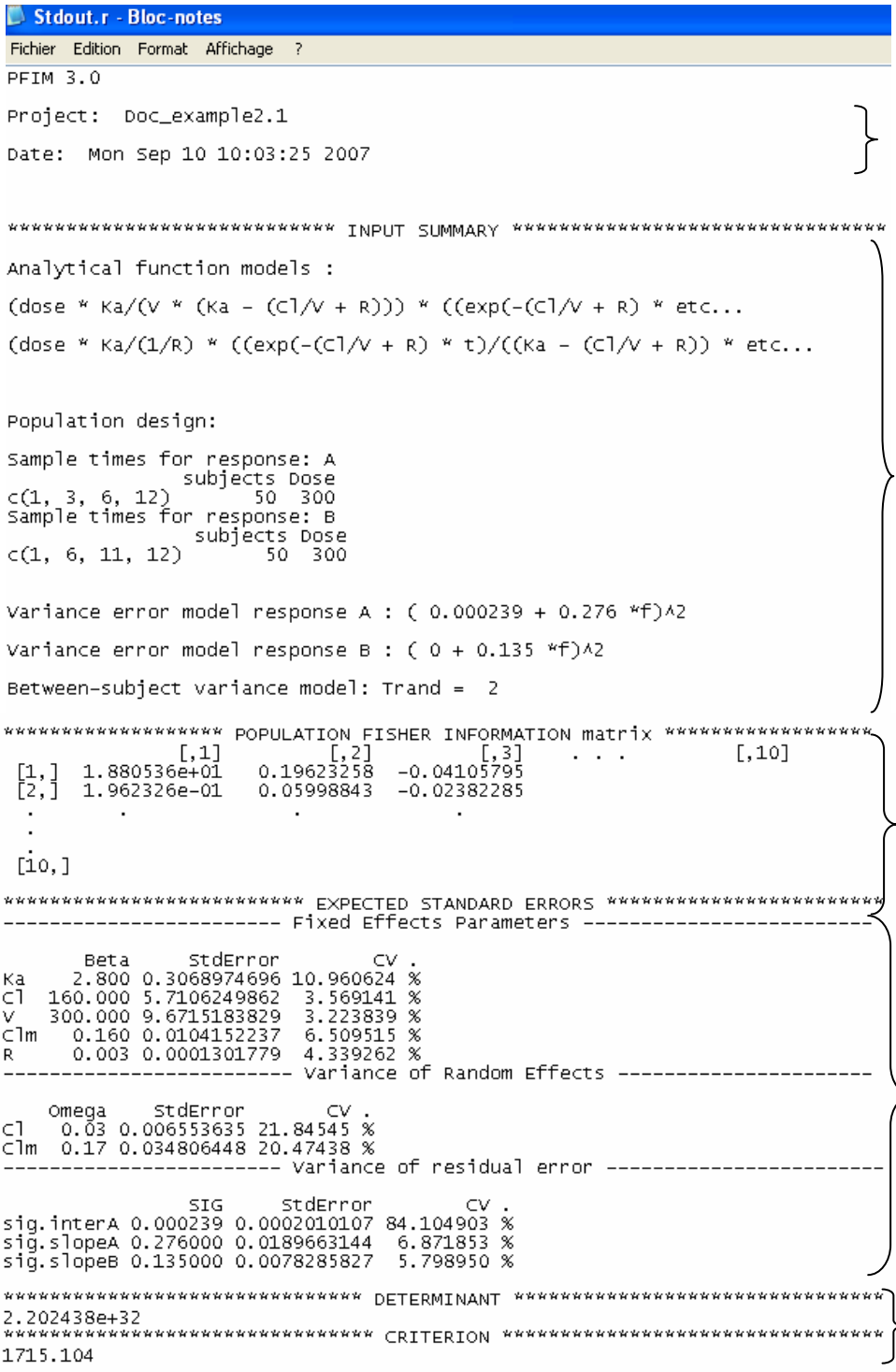


Figure 1. Example of design evaluation output file

Moreover, the PFIM() function returns the following R objects:

- **dose**
- **prot**: design evaluated for each response
- **subjects**: number of subjects for each group
- **mfisher**: the population Fisher information matrix
- **determinant**: the determinant of the population Fisher information matrix

- **crit**: the value of the criterion
- **p**: the vector
- **se**: the vector of the expected standard errors for each parameter
- **cv**: the corresponding coefficient of variation, expressed in persent.

5.2. Optimisation output file and objects

Figure 2 represents the output file from the design optimisation described in the Example documentation in the section 1.2.2. This example uses the Federov-Wynn algorithm.

The user can read on the Figure 2:

(1) The name of the function used: PFIM 3.0

(2) The name of the project and the date.

(3a) A summary of the input: model, variance error model, residual between-subject variance model, initial population design, initial numbers or proportions of subjects and doses.

(3b) Total number of allowed samples, criterion associated to the initial population design.

(3c) Sampling times specifications (according to the algorithm used) and error tolerances for the solver of differential equations system if used.

(4) The optimised design.

For the simplex algorithm, the number of iterations performed and the number of function evaluations are reported, so as the status of the convergence (false or achieved).

For the Federov-Wynn algorithm, the list of the allowed sampling times, the number of sampling points, the maximum and the minimum total number of points, the number of elementary designs evaluated for each response.

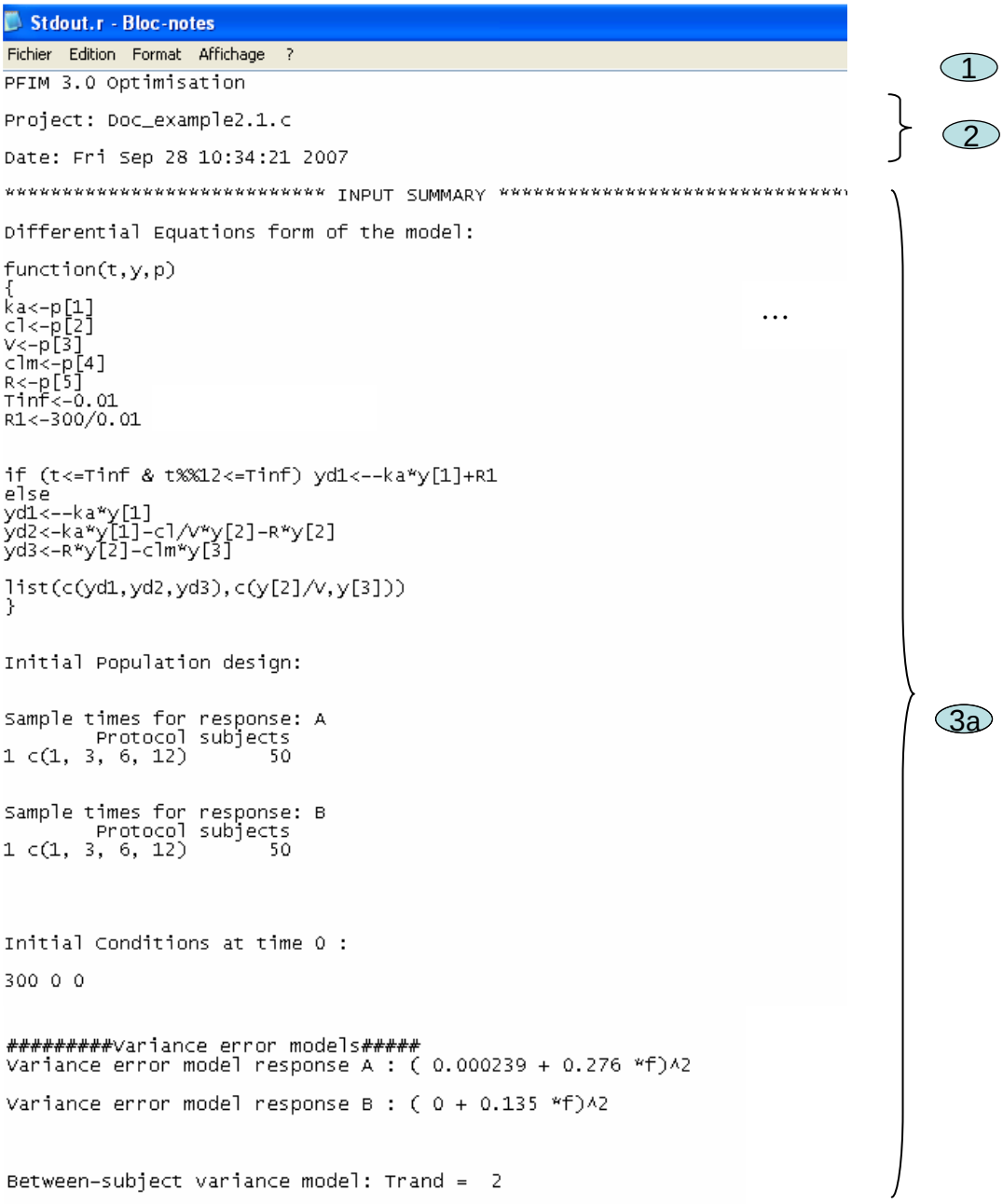
The value of the criterion associated with the optimised design is reported.

In this example, the design optimised is described by 3 groups with the same sampling time for each response. At each group corresponds a proportion of subjects optimised by the Federov-Wynn and the equivalence in number.

(5) The population Fisher information matrix, a dim*dim symmetric matrix where dim is the total number of population parameters to be estimated.

(6) The value of each population parameter with the expected standard error on each parameter and the corresponding coefficient of variation.

(7) The value of the determinant of the Fisher information matrix and the value of the criterion (determinant^(1/dim)) where dim denotes the number of population parameters.


Stdout.r - Bloc-notes
Fichier Edition Format Affichage ?
PFIM 3.0 Optimisation
Project: Doc_example2.1.c
Date: Fri Sep 28 10:34:21 2007
****************************** INPUT SUMMARY ********************************
Differential Equations form of the model:
function(t,y,p)
{
ka<-p[1]
cl<-p[2]
V<-p[3]
clm<-p[4]
R<-p[5]
Tinf<-0.01
R1<-300/0.01
if (t<=Tinf & t%%12<=Tinf) yd1<--ka*y[1]+R1
else
yd1<--ka*y[1]
yd2<-ka*y[1]-cl/V*y[2]-R*y[2]
yd3<-R*y[2]-clm*y[3]
list(c(yd1,yd2,yd3),c(y[2]/V,y[3]))
}
Initial Population design:
Sample times for response: A
Protocol subjects
1 c(1, 3, 6, 12) 50
Sample times for response: B
Protocol subjects
1 c(1, 3, 6, 12) 50
Initial Conditions at time 0 :
300 0 0
#########Variance error models#####
Variance error model response A : (0.000239 + 0.276 *f)^2
Variance error model response B : (0 + 0.135 *f)^2
Between-subject variance model: Trand = 2
...
1
2
3a

```
Total number of samples: 400

Associated criterion value: 3176.373
```

3b

```
Sampling windows for the response: A
Window 1 : t= 0.0625 1 3 6 11 12 14 15
    Nb of sampling points to be taken in this window, n[ 1 ]= 4
Maximum total number of points in one elementary protocol : 4
Minimum total number of points in one elementary protocol : 4


Sampling windows for the response: B
Window 1 : t= 0.0625 1 3 6 11 12 14 15
    Nb of sampling points to be taken in this window, n[ 1 ]= 4
Maximum total number of points in one elementary protocol : 4
Minimum total number of points in one elementary protocol : 4


Now evaluating the Fisher Information Matrix for the 70 protocols generated
Error tolerance for solving differential equations system: RtolEQ = 1e-08 , A
```

3c

```
****************************** OPTIMISED DESIGN ********************************


Optimised population design:
Sample times for response: A
             prot.opti subjects.opti Subjects
1  c(0.0625, 1, 3, 11)     0.2195818 10.97909
2 c(0.0625, 3, 11, 15)     0.4704995 23.52498
3  c(0.0625, 1, 6, 15)     0.3099187 15.49593


Sample times for response: B
             prot.opti subjects.opti Subjects
1  c(0.0625, 1, 3, 11)     0.2195818 10.97909
2 c(0.0625, 3, 11, 15)     0.4704995 23.52498
3  c(0.0625, 1, 6, 15)     0.3099187 15.49593

Associated optimised criterion: 4553.722
```

4

```
******************** POPULATION FISHER INFORMATION matrix ********************
               [,1]          [,2]          [,3]          [,4]          [,5]
 [1,]  3.929743e+02    0.13964935   -0.76222837 -5.103310e+00  9.263450e+05
 [2,]  1.396493e-01    0.04259301   -0.02428545  7.904761e-01 -7.735037e+02
 [3,] -7.622284e-01   -0.02428545    0.03226541 -1.618709e+00  1.596037e+03
 [4,] -5.103310e+00    0.79047609   -1.61870866  1.174554e+04 -8.917299e+05
 [5,]  9.263450e+05 -773.50371081 1596.03749598 -8.917299e+05  3.978996e+09
 [6,]  0.000000e+00    0.00000000    0.00000000  0.000000e+00  0.000000e+00
 [7,]  0.000000e+00    0.00000000    0.00000000  0.000000e+00  0.000000e+00
 [8,]  0.000000e+00    0.00000000    0.00000000  0.000000e+00  0.000000e+00
 [9,]  0.000000e+00    0.00000000    0.00000000  0.000000e+00  0.000000e+00
[10,]  0.000000e+00    0.00000000    0.00000000  0.000000e+00  0.000000e+00
              [,6]        [,7]          [,8]          [,9]        [,10]
 [1,]      0.00000    0.000000 0.000000e+00 0.000000e+00      0.00000
 [2,]      0.00000    0.000000 0.000000e+00 0.000000e+00      0.00000
 [3,]      0.00000    0.000000 0.000000e+00 0.000000e+00      0.00000
 [4,]      0.00000    0.000000 0.000000e+00 0.000000e+00      0.00000
 [5,]      0.00000    0.000000 0.000000e+00 0.000000e+00      0.00000
 [6,]  20350.28695    5.309660 1.376490e+05 3.433021e+02     80.50461
 [7,]      5.30966  859.978240 3.162648e+03 7.363451e+00     59.40044
 [8,] 137648.96027 3162.647932 1.148552e+09 1.599509e+05 39443.13910
 [9,]    343.30215    7.363451 1.599509e+05 2.842483e+03   112.18263
[10,]     80.50461   59.400443 3.944314e+04 1.121826e+02 15760.51146
```

5

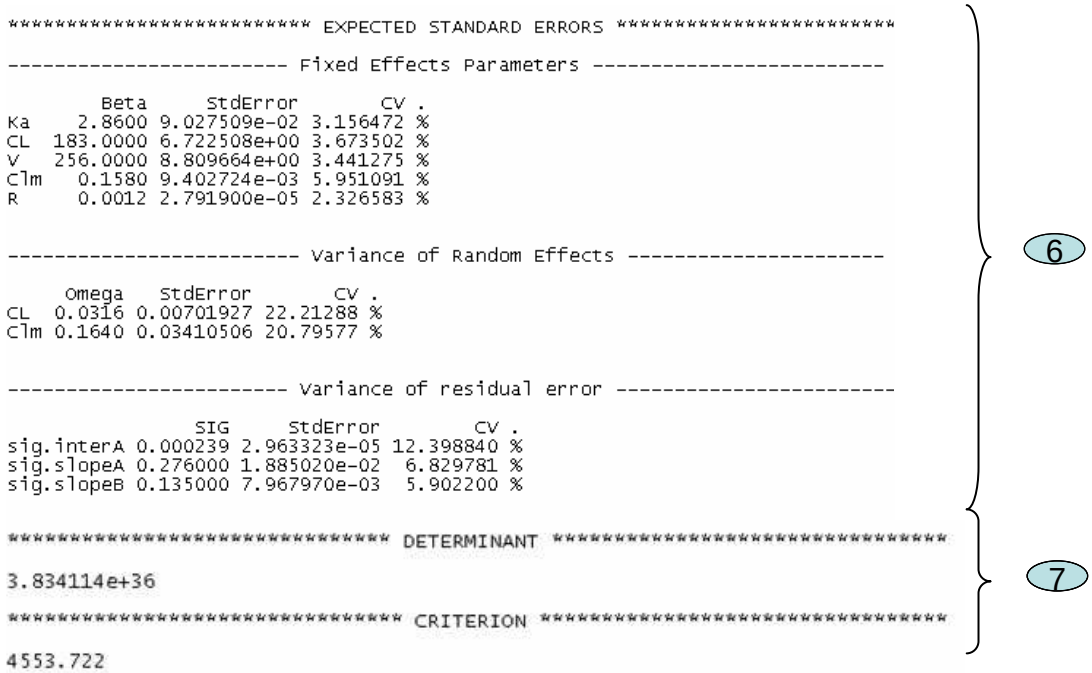


Figure 2. Example of design optimisation output file

Moreover, the PFIM() function returns the following R objects:

- **prot.opti**: the optimised design
- **subjects.opti**: the optimised proportion of subjects
- **mfisher**: the population Fisher information matrix
- **determinant**: the determinant of the population Fisher information matrix
- **crit**: the value of the criterion
- **p**: the vector of the parameters
- **se**: the vector of the expected standard errors for each parameter
- **cv**: the corresponding coefficient of variation, i.e. the se in percent.

6. Comments

o If the between-subject variance of a parameter is assumed to be zero, enter 0 for this variance in omega: PFIM will remove the corresponding row and column in the Fisher information matrix.

o If a population design leads to very poor information with a singular population Fisher information matrix (det=0), the expected standard errors and the coefficients of variation are returned as NA.

o For an infusion model, it is not possible for design evaluation to include the time at the end of the infusion when the end of infusion parameter as a parameter to be estimated.

o The number of subjects for each elementary design is the same for all the response(s).

o If the bound are not correctly specified according to the number of responses in case of analytical form, by default bounds are initialized to c(0, Inf) but that does not appear on the stdin.r file. In case of differential equation system, the bound is not used.

- In opposition to the last version, the user must put the variable “dose” in the analytical expression of the model and specify it in the input file if the user wants to define his model.

- An optimisation with an initial design composed of several groups with different doses (analytical form model) or initial conditions (ODE system model) is available with the Federov-Wynn alfgorithm (See example 2 in the optimisation part).