Population Stochastic Modelling (PSM): Model definition, description and examples

Stig Mortensen and Søren Klim

December 18, 2011

Package: PSM, version 0.8-6
URL: http://www.imm.dtu.dk/psm

Contents

1 Introduction

This package provides functions for estimation of linear and non-linear mixed-effects models using stochastic differential equations. Moreover it provides functions for finding smoothed estimates of model states and for simulation. The package allows for any multivariate non-linear time-variant model to be specified, and it also handels multidimentional input, co-variates, missing observations and specification of dosage regimen.

Some of the most essential parts of the implentation, namely the Kalman filter, is for linear models run using compiled code written in Fortran, which gives significant improvements in the parameter estimation times in R. However, otherwise this version is almost entirely created in *R*, and estimation times are thus in no way comparable to state-of-the-art software for similar types of models based on ordinary differential equations.

2 Model definition

A mixed-effects model is used to describe data with the following general structure

$$\boldsymbol{y}_{ij}, \quad i = 1, ..., N, \quad j = 1, ..., n_i \tag{1}$$

where $\boldsymbol{y}_{ij}$ is a vector of measurements at time t_{ij} for individual i, N is the number of individuals and n_i is the number of measurements for individual i. In a mixed-effects model the variation is split into intra-individual variation and inter-individual variation, which is modelled by a first and second stage model. For further detail regarding the model definition please refer to [1, 2, 3].

First stage model

The first stage model for a mixed effects model can be written in the form of a state space model. A state space model consists of two parts, namely a set of continuous state equations defining the dynamics of the system and a set of discrete measurement equations, which defines a functional relationship between the states of the system and the obtained measurements. In the linear form the state space equations are written as

$$\begin{aligned} d\boldsymbol{x}_t &= (\boldsymbol{A}(\boldsymbol{\phi}_i)\boldsymbol{x}_t + \boldsymbol{B}(\boldsymbol{\phi}_i)\boldsymbol{u}_t)dt + \boldsymbol{\sigma}_\omega(\boldsymbol{\phi}_i)d\boldsymbol{\omega}_t & (2)\\ \boldsymbol{y}_{ij} &= \boldsymbol{C}(\boldsymbol{\phi}_i)\boldsymbol{x}_{ij} + \boldsymbol{D}(\boldsymbol{\phi}_i)\boldsymbol{u}_{ij} + \boldsymbol{e}_{ij} & (3) \end{aligned}$$

and for a general non-linear model as

$$\begin{aligned} d\boldsymbol{x}_t &= \boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{u}_t, t, \boldsymbol{\phi}_i)dt + \boldsymbol{\sigma}(u_t, t, \phi_i)d\boldsymbol{\omega}_t & (4)\\ \boldsymbol{y}_{ij} &= \boldsymbol{g}(\boldsymbol{x}_{ij}, \boldsymbol{u}_{ij}, t_{ij}, \boldsymbol{\phi}_i) + \boldsymbol{e}_{ij} & (5) \end{aligned}$$

where t is the continuous time variable, the states of the model and the optional inputs at time t are denoted $\boldsymbol{x}_t$ and $\boldsymbol{u}_t$ respectively and $\boldsymbol{\omega}_t$ is a standard Wiener process such that $\boldsymbol{\omega}_{t_2} - \boldsymbol{\omega}_{t_1} \in N(\boldsymbol{0}, |t_2 - t_1|\boldsymbol{I})$. Both the state, measurement and input can be multi-dimensional, and are in such cases thus represented by a vector at time t_{ij}. The input is assumed constant between sample times (zero-order hold). The individual model parameters are denoted $\boldsymbol{\phi}_i$. Measurements are assumed observed with a Gaussian white noise measurement error, that is $\boldsymbol{e}_{ij} \in N(\boldsymbol{0}, \boldsymbol{S}(\boldsymbol{\phi}_i))$. For a non-linear model the covariance matrix may also depend on input and time, that is $S(\boldsymbol{u}_t, t, \boldsymbol{\phi}_i))$.

In the evaluation of the non-linear model it is necessary to specify a Jacobian matrix function with first-order partial derivatives for $\boldsymbol{f}$ and $\boldsymbol{g}$. These functions are defined as

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}_t} \quad , \quad \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}_t} \tag{6}$$

and must be given with the model specification. PSM will check the user defined Jacobian functions with numerical evaluations of the Jacobians of $\boldsymbol{f}$ and $\boldsymbol{g}$ to ensure that they are correct. It is possible to avoid specifying the Jacobian functions in the model and use numerical approximations instead, but this will increase estimation time at least ten-fold. See the help file in `R` for `PSM.estimate` for details regarding this.

The initial state of the model is given as a function of t_1, $\boldsymbol{\phi}_i$, and u_{i1} and defines the model state at time t_1 before update based on the first observation. The initial state can thus be included in the parameter estimation as necessary. The covariance matrix of the initial state is set to the integral of the Wiener process and the dynamics of the system over the first sample interval $t_2 - t_1$ as also done in [3].

The concept of states is essential to the understanding of the model setup. The state vector describes the state of the entire system and is only observable through measurement noise. The actual relation between measurements and states is defined in the measurement equation (3). A state can represent many different aspects of the system of interest, e.g. concentrations or amounts in compartments, a volume, a parameter with unknown time varying behavior or an input to the system, that we wish to estimate.

Second stage model

The second stage model describes the variation of the individual parameters $\boldsymbol{\phi}_i$ between individuals and is defined as

$$\boldsymbol{\phi}_i = \boldsymbol{h}(\boldsymbol{\theta}, \boldsymbol{\eta}_i, \boldsymbol{Z}_i) \tag{7}$$

where $\boldsymbol{\eta}_i$ is the multivariate random effect parameter for the ith individual, which is assumed Gaussian distributed with mean zero and covariance $\boldsymbol{\Omega}$, i.e. $\boldsymbol{\eta}_i \in N(\boldsymbol{0}, \boldsymbol{\Omega})$. The fixed effect parameter of the model is $\boldsymbol{\theta}$ and $\boldsymbol{Z}_i$ is a vector of co-variates for the ith individual.

3 Estimation

Parameter estimation is done using maximum likelihood. The likelihood function will only be outlined briefly here, so please refer to [1, 2, 3] for a detailed description.

The full set of model parameters to be estimated for the final mixed effects model based on SDEs are the matrices $\boldsymbol{\Sigma}$, $\boldsymbol{\sigma}_\omega$, $\boldsymbol{\Omega}$ and the fixed effect parameters in the vector $\boldsymbol{\theta}$. The three matrices are usually fixed to some degree so that only the diagonals or other partial structure remains to be estimated. In PSM the parameters in $\boldsymbol{\Sigma}$ and $\boldsymbol{\sigma}_\omega$ are included in $\boldsymbol{\theta}$.

In PSM the function `ModelPar` defines which part of the model parameters should estimated. These parameters are denoted $\boldsymbol{\Theta}$ (in PSM: `THETA`) such that

$$\texttt{ModelPar}: \quad \boldsymbol{\Theta} \rightarrow (\boldsymbol{\theta}, \boldsymbol{\Omega}). \tag{8}$$

The exact population likelihood function cannot be evaluated analytically and thus a second-order Taylor expansion is made of the individual a posteriori log-likelihood function around the value of $\hat{\boldsymbol{\eta}}_i$ that maximizes it. The objective function for PSM is thereby given as

$$-\log L(\boldsymbol{\Theta}) \approx \sum_{i=1}^{N} \left(\frac{1}{2} \log \left| \frac{-\boldsymbol{\Delta} l_i}{2\pi} \right| - l_i \right) \tag{9}$$

where l_i is the a posteriori log-likelihood function for the ith individual. This likelihood function is evaluated using the Kalman Filter which gives an exact solution for linear models. For non-linear models the Extended Kalman filter (EKF) is used which is only an approximation. The 2nd derivative $\boldsymbol{\Delta} l_i$ is approximated using the First-Order Conditional Estimation (FOCE) method, in the same way as it is normally done in mixed effects models based on ordinary differential equations (ODEs).

Uncertainty of parameters

PSM estimates the uncertainty for the parameter estimates based on the observed Fisher information. The parameters to be estimated are denoted $\boldsymbol{\Theta}$ and the observed information is then defined as

$$\boldsymbol{j}(\boldsymbol{\Theta}) = -\frac{\partial^2}{\partial \boldsymbol{\Theta} \partial \boldsymbol{\Theta}^T} \log L(\boldsymbol{\Theta}) = -\boldsymbol{\nabla}^2 \log L(\boldsymbol{\Theta}) \tag{10}$$

which is equal to the Hessian matrix of the negative log-likelihood function. If the parameters maximizing the likelihood function are called $\hat{\boldsymbol{\Theta}}$ they will asymptotically have the distribution

$$\hat{\boldsymbol{\Theta}} \sim N(\boldsymbol{\Theta}, \boldsymbol{j}(\hat{\boldsymbol{\Theta}})^{-1}). \tag{11}$$

This is used in PSM to provide a Wald 95% confidence interval, standard error and correlation matrix for the estimates. The Hessian is evaluated using `hessian` in the `numDeriv` package.

State estimates

A key feature of the SDE approach to population modelling is the ability to give improved estimates of the system states given the individual parameters and also to provide confidence bands for the states. Confidence bands at a time point t are directly given by the estimated state covariance matrix $\hat{\boldsymbol{P}}_{i(t|\ldots)}$ from the EKF, where t can be both at or between measurements.

There are four types of state and state covariance estimates available when using the EKF, each of which differs in the way data is used. The four types are:

- Simulation estimate: $\hat{\boldsymbol{x}}_{i(j|0)}$, $\hat{\boldsymbol{P}}_{i(j|0)}$
  Provides an estimate of the state evolution for a repeated experiment, without updating based on measurements. This is an ODE-like estimate, but it also yields a confidence band for the state evolution.

- Prediction estimate: $\hat{\boldsymbol{x}}_{i(j|j-1)}$, $\hat{\boldsymbol{P}}_{i(j|j-1)}$
  The prediction is used here to give the conditional density for the next observation at time t_{ij} given the observations up to $t_{i(j|j-1)}$.

- Filtering estimate: $\hat{\boldsymbol{x}}_{i(j|j)}$, $\hat{\boldsymbol{P}}_{i(j|j)}$
  Best estimate at time t_{ij} given the observations up to time t_{ij}.

- Smoothing estimate: $\hat{\boldsymbol{x}}_{i(j|N)}$, $\hat{\boldsymbol{P}}_{i(j|N)}$
  Optimal estimate at time t_{ij} utilizing all observations both prior to and after time t_{ij}.

For a conventional ODE model the state is found by the simulation estimate, which is entirely given by the (possibly ML-estimated) initial state of the system. The covariance matrix for the states is $\mathbf{0}$ since no system noise is estimated.

With SDEs three new types of estimates, apart from the simulation estimate, also become available. In the present setup the prediction estimate is used to give conditional Gaussian densities to form the likelihood function. The filter estimate is the best obtainable state estimate during the experiment, where the subsequent observations are not present. The third type of state estimate is the smoothed estimate. This provides the optimal state and state covariance estimate ($\hat{\boldsymbol{x}}_{i(j|N)}$ and $\hat{\boldsymbol{P}}_{i(j|N)}$) based on all obtained observations, both prior and subsequent to the time of interest. The smoothed estimate is therefore often the natural estimate of choice when studying the behavior of the system in *post hoc* analysis [1].

Comparison to NONMEM

The NONMEM software is a widely used tool for mixed effects modelling based on ODEs [4]. NONMEM also performs maximum likelihood estimation using the FOCE approximation and for many it might thus be of interest to know how the two objective functions are related.

The objective function in NONMEM (l_{NM}) is advertised as $-2\log L$ but in fact it lacks a constant equal to the likelihood of the data. The PSM objective function (l_{PSM}) is $-\log L$ as seen in Eq. (9) and the relation thereby becomes $l_{NM} = 2 \cdot l_{PSM} - \log(2\pi) \cdot \sum n_i$. The relation has been tested for a number of models [1].

Models with no random effects

A special case arises in models where no random effects are specified. This may be defined in PSM by setting `OMEGA = NULL` in the `ModelPar` function. This greatly simplifies the likelihood function, as it is no longer necessary to integrate

out the random effects. The likelihood function is thus reduced to a product of conditional probabilities of the observations. In the form of a log-likelihood function this may be written as

$$-\log L(\boldsymbol{\Theta}) = -\sum_{i=1}^{N}\sum_{j=1}^{n_i}\log p(y_{ij}|\cdot) \tag{12}$$

where the $\cdot$ indicates conditioning on parameters and past observations for individual i. The latter conditioning is necessary due to the inclusion of SDEs in the model.

The likelihood function in Eq. (12) is identical to the one used in CTSM [3], and is sometimes referred to as a pooled likelihood. The inner sum of the likelihood function is also equal to the a priori individual log-likelihood function in the mixed effects framework.

Implementation issues

The estimation algorithm in PSM for linear models is based on the ordinary Kalman filter, which has been written in Fortran for faster execution times. However, the Fortran code does not support a singular A-matrix, and will in these cases fall back on an R version of the Kalman filter. This may be circumvented by adding a very small value to the diagonal of A, at least in order to find the first rough parameter estimates.

4 User's guide to PSM

PSM is built around two key objects. These are

- a data object and
- a model object.

The data object contains sample times, observations and possible input, covariates and dosing regimen for all individuals and the model object contains everything related to the model.

Model object

Before setting up a model in PSM it is a good idea to write it down on paper and note the dimensions of the state, observations and possible input and random effects. When this is done the function `PSM.template()` can be used for both linear and non-linear models as shown below to print a template for the model.

```
> PSM.template(Linear=TRUE,dimX=3,dimY=1,dimU=0,dimEta=3)

MyModel <- vector(mode="list")
MyModel$Matrices=function(phi) {
  list(
       matA=matrix(c(  ), nrow=3, ncol=3),
       matC=matrix(c(  ), nrow=1, ncol=3)
       )
}
MyModel$h = function(eta,theta,covar) {
  phi <- theta
  phi
}
MyModel$S = function(phi) {
  matrix(c(  ), nrow=1, ncol=1)
}
MyModel$SIG = function(phi) {
  matrix(c(  ), nrow=3, ncol=3)
}
MyModel$X0 = function(Time,phi,U) {
  matrix(c(  ), nrow=3, ncol=1)
}
MyModel$ModelPar = function(THETA) {
  list(theta=list(  ),
       OMEGA=matrix(c(  ), nrow=3, ncol=3)
       )
}
```

The structure of the model object and what each function should return can be derived from the template shown above. It is important to keep the input arguments of all functions unchanged even though a particular model may not need use every argument in a function.

The input arguments to the functions are defined as follow:

Input arg.	Type
`THETA`	Vector (possibly named) containing the parameters to be maximum likelihood estimated.
`theta`	Vector of all parameters in the model.
`phi`	As standard a list of individual parameters, but can also be a vector as defined in `$h`.
`eta`	An unnamed vector of random effects.
`Time, time`	A scalar value.
`covar`	As specified in the user defined data object.
`x,u,U`	Column matrices with state or input at a sample time.

Data object

The data object is a list with one element for each individual in the data set. Each element in the list must contain:

- `Y` - Matrix with observations. Each columns holds one (possibly multi-dimensional) observation. Missing observations should be marked as NA.
- `Time` - Vector of sample times. The length must correspond to the number of columns in `Y` for the individual.

Each element in the list can optionally contain:

- `U` - Matrix with input at sample times. The input cannot contain missing values and is assumed constant between sample times (zero-order hold). It must have the same number of columns as `Y`.
- `covar` - A vector/list with covariates to be used in the function `$h`.
- `Dose` - A list containing three vectors: Time, State and Amount. See help(PSM.estimate) for more detail.

The data object is illustrated with a small example. The object shown contains 4 and 5 observations for two individuals sampled at different times and it also has 'BMI' as a covariate for each individual.

```
> MyData <- list()
> MyData[[1]] <- list(Time=1:4,Y=matrix(c(2.1,3.2,3.4,3.7),nrow=1),covar=c(BMI=20.1))
> MyData[[2]] <- list(Time=3:7,Y=matrix(c(1.9,2.1,2.0,2.9,3.5),nrow=1),covar=c(BMI=23.4))
```

Main functions

The PSM program is accessed in R through five main functions:

- `PSM.simulate(Model, Data, THETA, deltaTime)`
  Simulates data for multiple individuals. The number of individuals is determined by length(Data). The simulation is based on the Euler method to be able to simulate SDEs and thus a short time step should be chosen.

- `PSM.estimate(Model, Data, Par, CI)`
  Estimates population parameters for any linear or non-linear model. The Par argument is a list containing initial estimates and bounds for the parameter search.

- `PSM.smooth(Model, Data, THETA, subsample)`
  Optimal estimates of model states based on estimated parameters. It returns both the predicted, filtered and smoothed state estimates and for models with random effects an estimate of these are also returned.

- `PSM.plot(Data, Smooth, indiv, type)`
  Creates a matrix plot with a column for each individual. The rows can show observations, inputs, simulated and estimated states, residuals and auto-correlation functions. The x and/or y axis can be on log-scale and it is possible to list simulated or estimated random effects on the plot.

- `PSM.template(Linear,dimX,dimY,dimU,dimEta,file)`
  Creates a template with R-syntax to help setup a model in PSM. It works for both linear and non-linear models and it can output the resulting template to the screen or a file.

For detailed information please refer to the help files in R.

5 Examples

The following provides two examples to illustrate the use of PSM. It may also be useful when trying to set up a new model, by using the code shown as model templates. The document is written using Sweave[1], and thus all R-code in the examples shown here can be extracted into an R-script file by writing

```
> vignetteSrc = list.files(pattern = "PSM.Rnw",
+                     system.file("doc", package = "PSM"),
+                     full.names = TRUE)
> #vignetteSrc
> Stangle(vignetteSrc)

Writing to file PSM.R
```

in R. The file is saved in the current folder (`getwd()`) which allows for easy access to further experimentation with the code behind the examples.

To save time in the processing of the document all computer intensive operations are skipped by setting a flag `Redo = FALSE` and instead the outcome is loaded from a saved .Rdata file. Changing the flag to `TRUE` in PSM.Rnw and writing `Sweave("PSM.Rnw")` in R will generate a new version of the document with analysis and plots based on the new simulated data sets.

```
> Redo = FALSE
```

[1] http://www.ci.tuwien.ac.at/∼leisch/Sweave/

5.1 Dosing in two-compartment model (Linear)

This example illustrates how a standard two-compartment model with a random diffusion between the compartments can be set up. An overview of the model is shown in Figure 1.

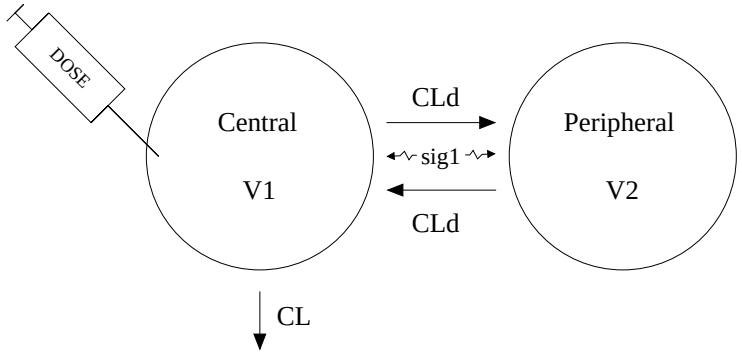

Figure 1: Model layout

The model is used to simulate data based on two doses of $1500mg$ given after 30 and 180 minutes. In state space formulation the model is described as

$$dA_1 = \left(-\frac{CL}{V_1^i}A_1 - \frac{CL_d}{V_1^i}A_1 + \frac{CL_d}{V_2}A_2\right)dt + \sigma_1 d\omega \quad (13)$$

$$dA_2 = \left(\frac{CL_d}{V_1^i}A_1 - \frac{CL_d}{V_2}A_2\right)dt - \sigma_1 d\omega \quad (14)$$

$$Y = A_1/V_1^i + e \quad (15)$$

or in matrix notation

$$d\boldsymbol{A}_t = \begin{bmatrix} -(CL/V_1^i + CL_d/V_1^i) & CL_d/V_2 \\ CL_d/V_1^i & -CL_d/V_2 \end{bmatrix}\boldsymbol{A}_t dt + \begin{bmatrix} \sigma_1 & 0 \\ -\sigma_1 & 0 \end{bmatrix} d\boldsymbol{\omega} \quad (16)$$

$$Y_{ij} = \begin{bmatrix} 1/V_1^i & 0 \end{bmatrix}\boldsymbol{A}_{ij} + e_{ij} \quad (17)$$

where $\boldsymbol{A}_t = [A_1 \ A_2]^T$ is the amount in each compartment and thus A_1/V_1^i is the measured concentration. It is seen that the elimination will follow a normal two-compartment model, but with a small random diffusion between the compartments. The mass is preserved since the diffusion terms are equal with opposite signs.

For simplicity the individual variation is modelled as

$$V_1^i = V_1 \exp(\eta_1) \quad (18)$$

The parameters of the model is $V_1 = 5L$, $V_2 = 10L$, $CL_d = 0.005L/min$, $CL = 0.002L/min$, $\sigma_1 = 10$, $S = 20mg^2/L^2$ and $\Omega = 0.5$.

The model can be defined in R as shown below.

```
> Model.SimDose = list()
> Model.SimDose$Matrices = function(phi) {
+   V1i <- phi$V1i; V2=phi$V2; CL = phi$CL; CLd = phi$CLd;
+   matA <- matrix(c(-(CL+CLd)/V1i ,  CLd/V2 ,
+                    CLd/V1i , -CLd/V2 ) ,nrow=2,byrow=T)
+   matC <- matrix(c(1/V1i,0),nrow=1)
+   list(matA=matA,matC=matC)
+ }
> Model.SimDose$X0 = function(Time=Na,phi,U=Na) {
+   matrix(0,nrow=2)
+ }
> Model.SimDose$SIG = function(phi) {
+   sig1 <- phi[["sig1"]]
+   matrix(c( sig1,0,
+             -sig1,0), nrow=2, byrow=T)
+ }
> Model.SimDose$S = function(phi) {
+   matrix(phi[["S"]])
+ }
> Model.SimDose$h = function(eta,theta,covar) {
+   phi <- theta
+   phi$V1i <- theta$V1*exp(eta[1])
+   phi
+ }
> Model.SimDose$ModelPar = function(THETA){
+   V2 <- 10
+   CLd <- 0.1
+   list(theta=list(V1 = THETA['V1'],V2=V2,CLd=CLd,CL=THETA['CL'], sig1=THETA['sig1'], S=THETA['S'])
+        OMEGA=matrix(THETA['OMEGA1']) )
+ }
> SimDose.THETA <-  c(CL=0.05,V1 = 5, sig1 = 10 , S = 20 , OMEGA1 = .2)
```

Five parameters in the model will be estimated, as it can be seen from the `ModelPar` function above. The parameters to be estimated are $\boldsymbol{\Theta} = (\texttt{CL}, \texttt{V1}, \texttt{sig1}, \texttt{S}, \texttt{OMEGA1})$.

For this example 5 individuals will be simulated. They will all be sampled every $10min$ for $400min$ which is described as below.

```
> N = 5
> SimDose.Data <- vector(mode="list",length=N)
> for (i in 1:N) {
+   SimDose.Data[[i]]$Time <- seq(from=10,by=10,to=400)
+   SimDose.Data[[i]]$Dose <-list(
+                                 Time = c(30,180),
+                                 State = c(1, 1),
+                                 Amount = c(1500,1500)
+                                 )
+ }
```

Everything is now setup and the simulation can be performed.

```
> if(Redo) {
+   SimDose.Data <- PSM.simulate(Model.SimDose, SimDose.Data, SimDose.THETA, deltaTime=.1)
+ } else
+   load("simdose.RData")
```

The simulated data are shown in Figure 2 using the `PSM.plot` function. The first row shows the observations for individuals 1 and 2, the next two show state 1 and 2 which we wish to estimate and the simulated values of η_1 is shown.

```
> PSM.plot(SimDose.Data,indiv=1:2,type=c('Y','longX','eta'))
> #par(mfcol=c(3,2),mar = c(2, 4, 2, 2)+.1)
> #for(id in 1:2) {
> #  plot(SimDose.Data[[id]]$Time , SimDose.Data[[id]]$Y,
> #         ylab="Observations", main=paste('Individual ',id,', eta= ',
> #                                  round(SimDose.Data[[id]]$eta,3),sep=""))
> #  for(i in 1:2) {
> #    plot(SimDose.Data[[id]]$longTime , SimDose.Data[[id]]$longX[i,],type="l",
> #         ylab=paste('State',i))
> #    rug(SimDose.Data[[id]]$Time)
> #  }
> #}
```

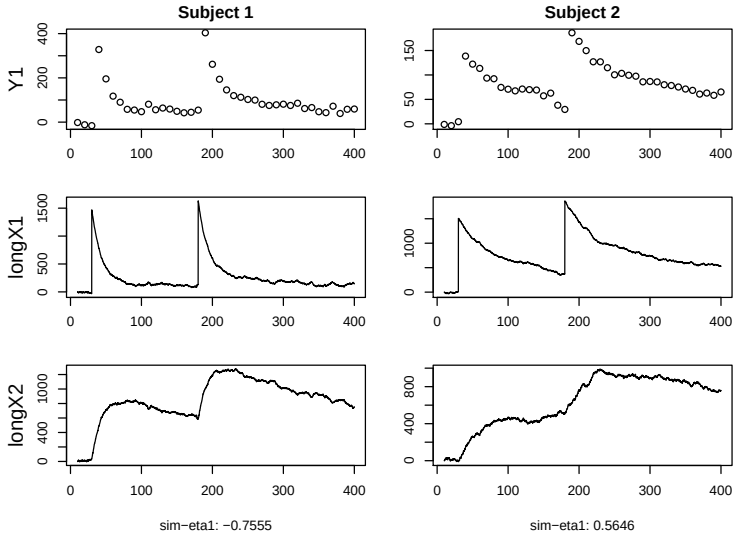

Figure 2: Simulated data and states.

As initial guess for the parameters in $\boldsymbol{\Theta}$ the true parameters are used and the bounds are $\pm$ a factor away.

```
> parA <- list(LB=SimDose.THETA*.5, Init=SimDose.THETA , UB=SimDose.THETA*1.5 )
> if(Redo) fitA <- PSM.estimate(Model.SimDose,SimDose.Data,parA,CI=T)
> fitA[1:5]

$NegLogL
[1] 725.8

$THETA
     CL      V1    sig1       S  OMEGA1
 0.0498  3.7223 10.2370 17.6984  0.2058

$CI
                CL    V1   sig1      S   OMEGA1
Lower CI95 0.04830 2.235  8.509  6.881 -0.03483
MLE        0.04980 3.722 10.237 17.698  0.20576
Upper CI95 0.05129 5.210 11.965 28.515  0.44635

$SD
            CL     V1   sig1     S OMEGA1
[1,] 0.0007615 0.7589 0.8816 5.519 0.1227

$COR
              CL        V1     sig1         S    OMEGA1
CL      1.000000 -0.007664  0.01799 -0.015349 -0.009738
V1     -0.007664  1.000000  0.01224  0.001579 -0.001153
sig1    0.017992  0.012238  1.00000 -0.557870 -0.021768
S      -0.015349  0.001579 -0.55787  1.000000  0.022274
OMEGA1 -0.009738 -0.001153 -0.02177  0.022274  1.000000

> SimDose.THETA

    CL     V1   sig1      S OMEGA1
  0.05   5.00  10.00  20.00   0.20
```

It is seen that the estimated parameters are reasonably close to the true values in `SimDose.THETA` and the 95% confidence intervals include the true values. In particular the first parameter σ_1 is significantly different from zero which shows that deviations from a normal ODE two-compartment model are significant.

Based on the estimated parameters it is possible to obtain an estimate of the model states by using `PSM.smooth`. The estimates are shown in Figure 3. The structure of the output from the smoothing function is also shown using the `names` command. Please refer to `help(PSM.smooth)` for a more detailed description of the output.

```
> if(Redo)
+   out <- PSM.smooth(Model.SimDose, SimDose.Data, fitA$THETA, subsample = 20)
> # View the data structure
> names(out[[1]])

 [1] "Time"    "Xs"      "Ps"      "Ys"      "Xf"
 [6] "Pf"      "Xp"      "Pp"      "Yp"      "R"
[11] "eta"     "negLogL"
```

By comparing the smoothed estimates of the states to the true simulated states, it can be seen that they are very close. This shows that the system noise and observation noise has been separated properly in the reconstruction.

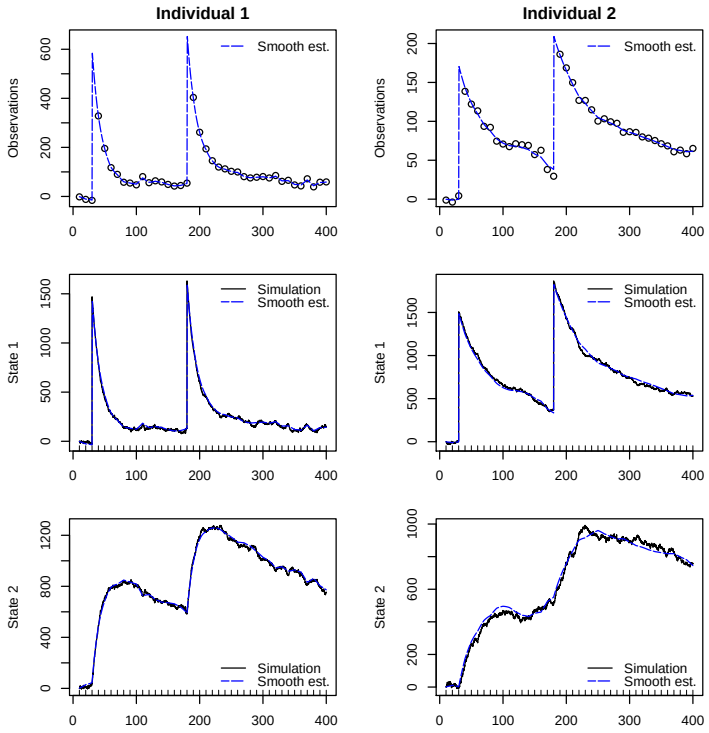

Figure 3: Smoothed estimate of states.

5.2 Extraction of insulin secretion rate (Linear)

Insulin secretion rates (ISR) can be estimated based on measurements of the concentration of C-peptide in the blood, since insulin and C-peptide are secreted in equi-molar amounts. This example will first illustrate a way to simulate C-peptide data based on a model for ISR, and then how ISR can be estimated again using a more simple model. The models used in the example are described in further detail in [5].

The simulated measurements of C-peptide spans over 24H, during which the patients recieves three meals at 8 a.m., 12 a.m. and 6 p.m. These meals will give rise to an increase in insulin secretion, which will be modelled and estimated.

The simulation model for the C-peptide measurements is based on the commonly used two compartment model for C-peptide as shown in Figure 4. The kinetic parameters are set equal to the Van Cauter estimates [6].

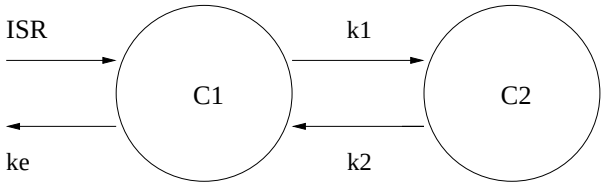

Figure 4: Model layout

The first two states of the simulation model is concentration in compartment 1 and 2, $C1$, $C2$. The third state is ISR and is the secretion which is modelled as a structural part based on the three meal time plus a constant baseline B and random noise through a Wiener process. The fourth state Q is used to model ISR, where Q is controlled by an input u_2 which is equal to 1 for $30min$ after meal times. The model can be defined as

$$\begin{aligned} dC_1 &= [-(k_1 + k_e)C_1 + k_2 C_2 + ISR]dt & (19)\\ dC_2 &= [k_1 C_1 - k_2 C_2]dt & (20)\\ dISR &= [-a_1 ISR + a_1 Q + B^i]dt + \sigma_{ISR} d\omega & (21)\\ dQ &= [-a_2 Q + a_2 K^i u_2]dt & (22) \end{aligned}$$

which is linear and can thus again be written on matrix form.

The model is initialized in steady state just prior to the first meal time. The individual variation in the model is included in the initial concentration $C_1(0)^i$, baseline B^i and height of the peaks K^i such that

$$\begin{aligned} B^i &= B\exp(\eta_1) & (23)\\ K^i &= K\exp(\eta_2) & (24)\\ C_1(0)^i &= C_1(0)\exp(\eta_3) & (25) \end{aligned}$$

In order to write a model containing a constant in the differential equations (here B^i) on the linear form as defined in Eq. (2) and (3) it is necessary to include a constant input $u_1 = 1$ and multiply this with B^i. The matrix description of the model can be found in [5] p. 69. Using this, the model can be defined in PSM as follows.

```
> k1 = 0.053; k2 = 0.051; ke = 0.062;
> Model.SimISR <- list()
> Model.SimISR$Matrices = function(phi) {
+   a1  <- phi[["a1"]]
+   a2  <- phi[["a2"]]
+   B  <- phi[["B"]]
+   K  <- phi[["K"]]
+   matA <- matrix( c(-(k1+ke) ,  k2 ,   1 ,   0,
+                     k1 , -k2 ,   0 ,   0,
+                     0 ,   0 , -a1 ,  a1,
+                     0 ,   0 ,   0 , -a2),nrow=4,byrow=T)
+   matB <- matrix( c(0 , 0   ,
+                     0 , 0   ,
+                     B , 0   ,
+                     0 , a2*K),byrow=T,nrow=4)
+   matC <- matrix(c(1,0,0,0),nrow=1)
+   matD <- matrix(c(0,0),nrow=1)
+   list(matA=matA,matB=matB,matC=matC,matD=matD)
+ }
> Model.SimISR$X0 = function(Time=NA,phi,U=NA) {
+   C0 <- phi[["C0"]]
+   tmp    <- C0
+   tmp[2] <- C0*k1/k2
+   tmp[3] <- C0*ke
+   tmp[4] <- 0
+   matrix(tmp,ncol=1)
+ }
> Model.SimISR$SIG = function(phi) {
+   diag( c(0,0,phi[["SIG33"]],0))
+ }
> Model.SimISR$S = function(phi) {
+   return( matrix(phi[["S"]]))
+ }
> Model.SimISR$h = function(eta,theta,covar) {
+   phi <- theta
+   phi[["B"]] <- theta[["B"]]*exp(eta[1])
+   phi[["K"]] <- theta[["K"]]*exp(eta[2])
+   phi[["C0"]] <- theta[["C0"]]*exp(eta[3])
+   return(phi)
+ }
> Model.SimISR$ModelPar = function(THETA){
+   list(theta=list(C0=900,S=8500,
+                   a1=THETA['a1'],a2=THETA['a2'],
+                   SIG33=THETA['SIG33'],
+                   K = THETA['K'], B = THETA['B']),
+                OMEGA=diag(c(.2,.2,.2))
+        )
+ }
>
```

For this example two individuals will be simulated. They will both be sam-

pled at predefined time points during 24H. This defined as below together with the input data for each individual.

```
> Sim.Data <- vector(mode="list",length=2)
> for (i in 1:2) {
+   Sim.Data[[i]]$Time <- c( 0,15,30,45,60,75,90,120,150,180,210,240,270,300,330,
+                            360,420,480,600,615,630,645,660,675,690,720,750,780,810,
+                            840,960,1140,1320,1410,1440)
+   Sim.Data[[i]]$U <- matrix(c( rep(1,35) ,
+                                as.numeric( Sim.Data[[i]]$Time %in% c(0,15,240,600,615)) )
+                             ,byrow=T,nrow=2)
+ }
```

Both the model, sample times and input is now prepared and the simulation can be performed. The parameter estimates are taken from [5] p. 70. The simulated data are shown in Figure 5.

```
> Sim.THETA <-  c(a1=0.02798, a2=0.01048, SIG33=4 , K=427.63 , B=1.7434)
> if(Redo) {
+   Sim.Data <- PSM.simulate(Model.SimISR, Sim.Data, Sim.THETA, deltaTime=.1 )
+ } else
+   load("simisr.RData")
```

The next step is to generate a model for estimation of ISR. It is again based on the model illustrated in Figure 4 only this time the ISR is simply modelled as a random walk. Thus no information about the meal times is used in the estimation of ISR.

The model simplification is done by replacing Eq. (21) and (22) by Eq. (26) below. The model for estimation can thus be seen as estimating the outcome of the random walk for ISR based on the observed (simulated) data for $C1$.

$$dISR = \sigma_{ISR} \tag{26}$$

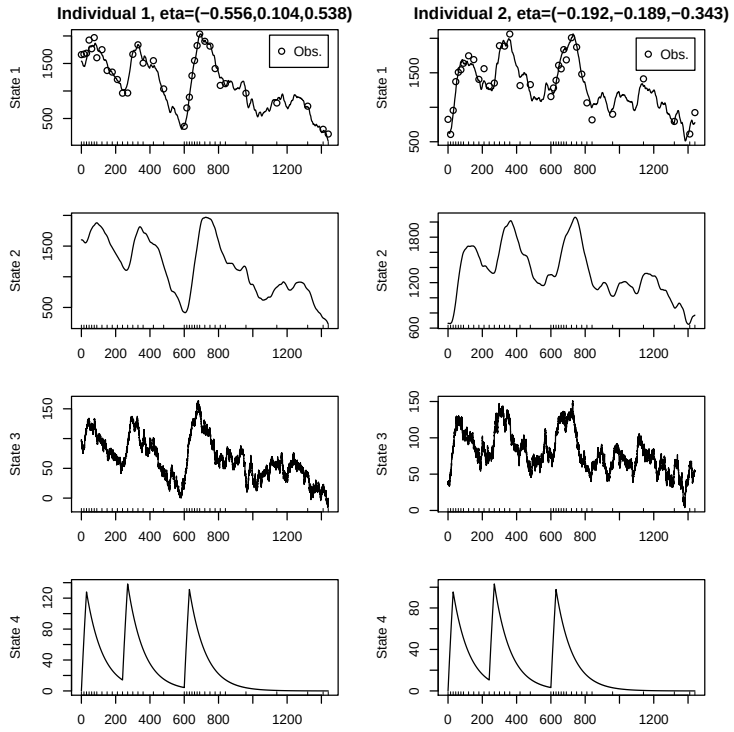

Figure 5: Simulated data and states.

```
> Model.Est <- list(
+             Matrices=function(phi) { list(
+               matA=matrix(c(-(k1+ke),  k2, 1,
+                             k1      , -k2, 0,
+                             0       ,   0, 0  ),ncol=3,byrow=T),
+               matB=NA,
+               matC=matrix(c(1,0,0),nrow=1),
+               matD=NA )  },
+             X0 = function(Time=NA,phi=NA,U=NA) {
+               C0 <- phi[["C0"]]
+               tmp    <- C0
+               tmp[2] <- C0*k1/k2
+               tmp[3] <- C0*ke
+                     return(matrix(tmp,ncol=1) )} ,
+             SIG = function(phi) {
+                      return( diag( c(1e-3,1e-3,phi[["SIG33"]])) ) } ,
+             S = function(phi) {
+                      return( matrix(phi[["S"]])) } ,
+             h = function(eta,theta,covar) {
+               phi <- theta
+               phi[["C0"]] <- theta[["C0"]]*exp(eta[1])
+               return(phi) } ,
+             ModelPar = function(THETA){
+               return(list(theta=list(C0=THETA['C0'],S=THETA['S'],SIG33=THETA['SIG33']),
+                           OMEGA=matrix(THETA['OMEGA'])))}
+             )
```

Looking at the `ModelPar`-function it is seen that it is chosen to include the average initial concentration $C_1(0)$, measurement variation S, the coefficient of the random walk for ISR σ_{ISR} and the variance of the random effect on $C_1(0)$ denoted $\Omega_{C_1(0)}$ in the likelihood estimation.

Since the model now does not use any input, this must be removed from the simulated data before estimation.

```
> Pop.Data <- Sim.Data
> for (i in 1:2)
+   Pop.Data[[i]]$U <- NULL
```

The data and model for estimation is now prepared, and the model can be estimated by calling `PSM.estimate`. This is done below and the output in `obj1[1:3]` containing the log-likelihood value, parameter estimates and confidence intervals is shown as output.

```
> par1 <- list(LB   = c(C0= 200, S= 50^2, SIG33= 0, OMEGA=.0 ),
+              Init = c(C0=1000, S=100^2, SIG33=10, OMEGA=.25),
+              UB   = c(C0=3000, S=150^2, SIG33=15, OMEGA=.50))
> if(Redo) obj1 <- PSM.estimate(Model.Est, Pop.Data, par1,CI=T,trace=1)
> obj1[1:5]

$NegLogL
[1] 497.6

$THETA
       C0         S     SIG33     OMEGA
1.121e+03 1.023e+04 4.768e+00 1.464e-01

$CI
                C0     S SIG33   OMEGA
Lower CI95  490.8  2265 3.556 -0.1717
MLE        1121.1 10227 4.768  0.1464
Upper CI95 1751.5 18188 5.979  0.4645

$SD
        C0    S  SIG33  OMEGA
[1,] 321.6 4062 0.6181 0.1623

$COR
             C0        S    SIG33    OMEGA
C0     1.00000 -0.03063  0.03367 -0.09409
S     -0.03063  1.00000 -0.39152  0.04218
SIG33  0.03367 -0.39152  1.00000 -0.05177
OMEGA -0.09409  0.04218 -0.05177  1.00000
```

Looking at the estimated confidence intervals, it is seen that the values used in the simulation $\boldsymbol{\Theta}$ = (900,8500,4,0.2) are nicely contained within the limits.

The estimation time including the confidence interval is about 3 minutes on a 2GHz computer. Since the matrix A in the estimation model is singular,

the estimation cannot make use of the compiled Fortran code. As mentioned, this may be circumvented by adding e.g. 10^{-6} to the diagonal. This reduces the estimation time to 12 sec. It changes the maximum log-likelihood value to 499.4781, and thus yields virtually no difference in parameter estimates.

Using the estimated model parameters it is possible to give smoothed estimates of the three model states $C1$, $C2$ and ISR. This is done below and the result is plottet in Figure 6. In Figure 7 the smoothed ISR state is plotted together with the estimated uncertainty. For both figures the true simulated states is also plotted for reference.

```
> if(Redo)
+   Data.Sm <- PSM.smooth( Model.Est , Pop.Data, obj1$THETA, subsample=10)
```

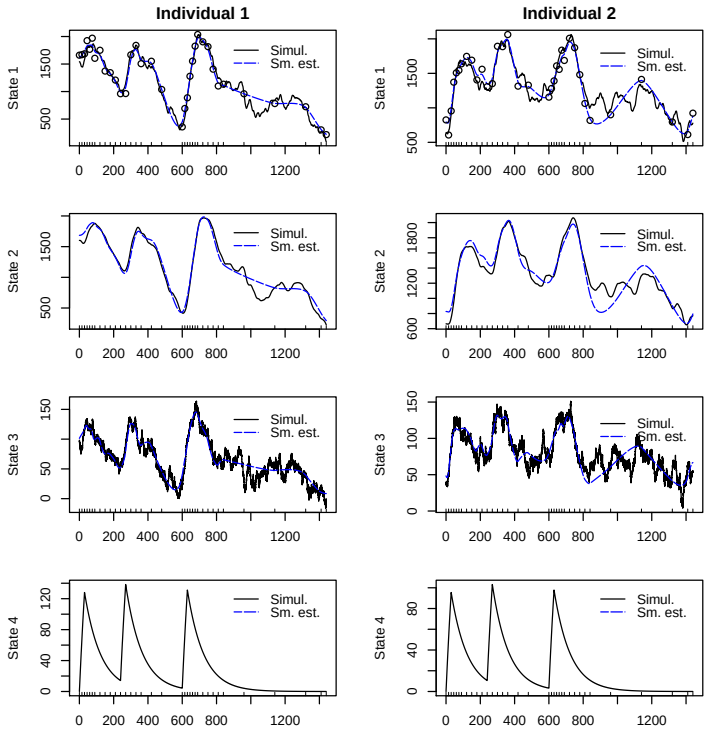

Figure 6: Smoothed estimate of states.

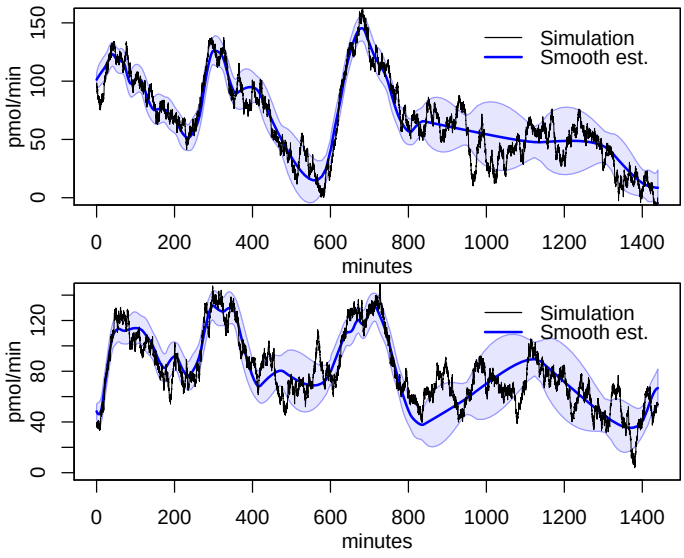

Figure 7: Smoothed estimate of insulin secretion rate ±1SD for individual 1 and 2 compared with the true simulated ISR.

References

[1] Mortensen SB, Klim S, Dammann B, Kristensen NR, Madsen H, Overgaard RV (2007) A Matlab framework for estimation of nlme models using stochastic differential equations: applications for estimation of insulin secretion rates. J of Parmacokinet Pharmacodyn 34:623-642

[2] Overgaard RV, Jonsson N, Tornøe CW, Madsen H (2005) Non-linear mixed-effects models with stochastic differential equations: implementation of an estimation algorithm. J of Parmacokinet Pharmacodyn 32(1):85-107

[3] Kristensen NR, Madsen H (2003) Continous time stochastic modelling: CTSM 2.3 mathematics guide, Technical University of Denmark
http://www2.imm.dtu.dk/ctsm/MathGuide.pdf

[4] Beal SL, Sheiner LB (2004) NONMEM®Users Guide. University of California, NONMEM Project Group.

[5] Klim S, Mortensen SB (2006) Stochastic PK/PD Modelling. M.Sc. thesis, Informatics and Mathematical Modelling, Technical University of Denmark.
http://www2.imm.dtu.dk/pubdb/views/edoc_download.php/4533/pdf/imm4533.pdf

[6] Van Cauter E, Mestrez F, Sturis J, Polonsky KS (1992) Estimation of insulin secretion rates from C-peptide levels. Comparison of individual and standard kinetic parameters for C-peptide clearance. Diabetes, 41(3), pp. 368-77.