

Supplementary Annex to

Statistical mechanical approach to incremental parameter evaluation from incomplete data with application to the population pharmacology of anticoagulants

M.O. Vlad, A.D. Corlan, F. Moran, P. Oefner, J. Ross

---

## Table of Contents

---

### **1 Purpose**

### **2 Overview**

### **3 Population PK/PD model**

#### 3.1 Individual PK/PD model

##### 3.1.1 The intestinal compartment

##### 3.1.2 The body compartment

##### 3.1.3 Interindividual variability parameters

#### 3.2 Population parameters

##### 3.2.1 Naming convention

##### 3.2.2 Genotype-related parameters

##### 3.2.3 Distribution of anthropometric parameters and other parameters

#### 3.3 Monte Carlo generation of individual constants

#### 3.4 The PK/PD simulator

### **4 Simulation result functionals and calibration**

#### 4.1 Functionals on the case space

#### 4.2 Calibration experiments

##### 4.2.1 Basic checks

##### 4.2.2 Calibration procedures

### **5 Bayesian experiment**

#### 5.1 Variables

#### 5.2 Procedure

### **6 Output**

#### 6.1 Output switch variables

#### 6.2 General output procedures

### **7 System integration**

#### 7.1 System options

#### 7.2 Utility function and procedures

#### 7.3 Function and procedure overview

#### 7.4 Compilation units overview

### **8 Graphics and calibration report**

### **9 Technical issues**

#### 9.1 System dependencies

#### 9.2 Usage

### **10 References**

### **11 Results**

---

# 1 Purpose

The purpose of the code below is to illustrate the application of the bayesian method for using INR measurements to improve prediction of the risk of out of range INR values for a given drug schedule in a population model of warfarin PK/PD.

## 2 Overview

A set of presumably independent individual parameters known to be strongly associated with variability of patient sensitivity to warfarin [1,2] have been modeled as a random vector. `Case_Fill` elements of the vector (cases) are randomly generated in the `Db` array. A drug schedule (here, a fixed simulated dose of 5 mg of warfarin every 24 hours) is assigned to each simulated case (`Generate_Constant_Daily_Dose`).

For each case, `I`, the INR at every hour (INR trace) is calculated using a system of ordinary differential equations where the above mentioned random variables appear as constant coefficients. This is performed by the `Calculate_All_Trajectories` procedure; results are stored in `Db(I).Inr` for each case.

The relative density of these values, at a given instant in time, estimates the instantaneous population INR response at that instant.

In procedure `Bayes_Correction_Experiment_MC`, one of the generated cases is picked at random (local variable `Simulated`). A set of INR measurements, `Datam`, is randomly generated by adding gaussian noise and resampling at a series of instants `Datai`. (`Datai,Datam`) pairs are used in successive bayesian steps. The prior distribution of the parameters is represented by associating the Monte Carlo generated parameters of each case with the initially equal probabilities in the `Pii` vector. The posterior distribution of each step is calculated by changing the `Pii` values and is used as a prior probability distribution for the next step.

The output of the program consists of a set of data files with the INR traces, the current probability of each INR value at each hour of simulated time, simulated measurements, case parameters and calibration output. Calibration output consists of tables and graphics of features in samples of the generated population (such as average daily warfarin dosage) that attempt to replicate graphics and tables from the experimental literature on samples from actual human populations.

## 3 Population PK/PD model

Our population PK/PD model includes: (A) a stochastic model of parameter distribution in the population, consisting of independent real random variables and (B) a system of deterministic ordinary differential equations, named the individual PK/PD model. These equations correspond to the kinetics of absorption and elimination of the drug and of the enzymatic reactions that lead to the reduction of coagulation factor synthesis rate. Random variables from the stochastic model appear as coefficients in the individual PK/PD model and are constant for a given individual case.

### 3.1 Individual PK/PD model

We use a two-compartment model, a compartment representing the digestive tract and the second representing the rest of the body. As the sole purpose of this example is to illustrate the bayesian method we used the simplest possible representation of the reaction kinetics involved; we performed basic calibration so that general features of the INR response to warfarin, as described in the cited literature, are reproduced by our model.

### 3.1.1 The intestinal compartment

Drug administration is modeled as a train of  $\delta$  functions (the **Q** component of **Case\_Model**). The rate of absorption is a rectangular signal that is non-zero (and proportional to the input dose) for **AbsorptionDuration** hours after administration and zero in the rest of the time.  $F_{in}(t)$  below represents this signal (rate of absorption). This represents the fact that warfarin is quickly and completely absorbed from the intestine [3] and provides results very close to more complex models in this case. Initialization of the drug administration signal is done by **Generate\_Constant\_Daily\_Dose**.

```
Generate constant daily dose[1] ≡
{
    procedure Generate_Constant_Daily_Dose(Dose: Float:= 1.0; From_Day: Float:= 0.0)
is
begin
    for I in 1..Case_Fill loop
        for J in 1..Schedule_Max loop -- could optimize by not wasting schedule
space
            Db(I).Q(J).T := 24.0 * (Float(J) + From_Day);
            Db(I).Q(J).D := Dose / (GDISTF * Db(I).C.Weight ** GWGPOW);
        end loop;
    end loop;
end Generate_Constant_Daily_Dose;
}
```

This macro is invoked in definition 26.

### 3.1.2 The body compartment

The kinetic equations for our model of the PK/PD of warfarin are:

$$\frac{dF_R(t)}{dt} = F_{in}(t) - c_{er}F_R(t) \quad (S1)$$

$$\frac{dF_S(t)}{dt} = F_{in}(t) - c_{es}F_S(t) \quad (S2)$$

$$\frac{dF_V(t)}{dt} = c_v c_{v_0} - c_v F_V(t) - c_s F_S(t) F_V(t) - c_r F_R(t) F_V(t) \quad (S3)$$

$$\frac{dF_H(t)}{dt} = -\frac{dF_O(t)}{dt} = F_O(t)(c_d F_V(t) + c_i) - c_a F_H(t) \quad (S4)$$

$$\frac{dF_2(t)}{dt} = c_2 c_{2i} F_H(t) - c_2 F_2(t) \quad (S5)$$

$$\frac{dF_7(t)}{dt} = c_7 c_{7i} F_H(t) - c_7 F_7(t) \quad (S6)$$

$$\log I(t) = a_2 \log F_2(t) + a_7 \log F_7(t) \quad (S7)$$

Here terms followed by  $(t)$  are real time functions while terms named  $c$  or  $a$  are either Monte Carlo generated individual case constants or population-wide constants, the same for all cases.

- $F_R(t)$ ,  $F_S(t)$  are the instantaneous level of the R- and S- enantiomers of warfarin.
- $F_{in}(t)$  is the rate of absorption, mentioned above.

- $c_{er}$ ,  $c_{es}$  are the rates of elimination of the R and S enantiomers; they depend in different ways on the activity level of the CYP2C9 enzyme, are constant for a given case and are produced by `Generate_Random_Cases` for the whole population from two distributions that are each mixture of six gaussian distributions corresponding to the most prevalent genotypes of CYP2C9 (\*1\*1, \*1\*2, \*1\*3, \*2\*2, \*2\*3 and \*3\*3) using the `CYP2C9 population parameters` below.
- $F_V(t)$  is the activity level of the VKORC1 enzyme.
- $c_v$  is constant for the individual and randomly generated for the population; a mixture of three gaussians is used for the AA, GA and GG genotypes, as designated in [2] based on the `VKOR population parameters` below and reflects the variable level of VKORC1 synthesis in the population.
- $c_{v_0}$  is 1.
- $c_r$ ,  $c_s$  are population-wide reaction constants of the VKOR with warfarin enantiomers and are constant (reflecting the generally identical structure of VKOR in the population).
- $F_O(t)$ ,  $F_H(t)$  represent the level of epoxyde and reduced vitamin-K respectively; their sum is constant in time for an individual in this model (absorption and elimination of vitamin-K is not represented here) and is the  $c_k$  individual constant that does not appear in the equations above and is randomly generated from `vitamin K pool parameters` with a gaussian distribution.
- $c_d$ ,  $c_i$  and  $c_a$  are population-wide constants representing, respectively, the rate of VKOR-dependent and VKOR-independent reduction of vitamin-K epoxyde and the rate of oxydation with  $\gamma$ -carboxylation; here we assume that  $\gamma$ -carboxylation of other substrates is much more important for vitamin-K oxydation than  $\gamma$ -carboxylation of coagulation factors.
- $F_2(t)$  and  $F_7(t)$  are circulating levels of coagulation factors II and VII.
- $c_2$  and  $c_7$  are population-wide constants that reflect the turnover of coagulation factors II and VII.
- $c_{2i}$  and  $c_{7i}$  are individually constant levels of protein precursors of coagulation factors II and VII available for  $\gamma$ -carboxylation that are generated with a gaussian distribution corresponding to baseline levels in a healthy european population [4].
- $I(t)$  is the international normalized ratio (INR) of the prothrombin time for the case—the indicator used to measure treatment effect.
- $a_2$  and  $a_7$  are population-wide constants that relate instantaneous factor levels to the INR, empirically determined as reported in [4].

### 3.1.3 Interindividual variability parameters

Identifiers of constants from the system of equations appear below. `Fin`, that corresponds to the rectangular signal of the absorption rate, appears here as a constant because it doesn't change during a time step in the numerical simulations. The population-wide constants will be set from values produced by `Generate_Random_Cases`.

*parameters of the system of equations*[2]  $\equiv$

{

```

Fin: Float:= 0.0;
Cer: Float:= 0.2;
Ces: Float:= 0.9;
Onev: Float:= 1.0;
Cv: Float:= 0.05;
Cs: Float:= 2.43;
Cr: Float:= 0.012;

```

```

Ca:   Float:= 0.12;
Ck:   Float:= 2.0;
Cd:   Float:= 0.37;
Ci:   Float:= 0.001;
C7:   Float:= 0.48 / 6.0;
C10:  Float:= 0.48 / 24.0;
C2:   Float:= 0.48 / 32.0;
C7i:  Float:= 1.3;
C2i:  Float:= 1.33;
C10i: Float:= 1.33;

```

```

}
```

This macro is invoked in definition 11.

*other individual phenotypic parameters*[3] ≡

```

{
```

```

Weight: Float:= 80.0;      -- kg
Gender: Gender_Type:= Male;
Age:    Float:= 50.0;      -- years
Height: Float:= 1.80;      -- meters
Surface: Float:= 2.0;      -- body surface sq meters
Absorption_Duration: Float:= 1.0; -- hours
Blood_Volume: Float:= 6.0;  -- liters

```

```

}
```

This macro is invoked in definition 11.

## 3.2 Population parameters

Parameters that describe the population distribution of individual variables are enumerated below. We used mixtures of gaussian distributions described as sets of (mean, dispersion, prevalence) triplets.

### 3.2.1 Naming convention

The suffix of parameter names below is MX if the constant is a maximum value, MI if it is a minimum, ME for mean, SD for dispersion and FR for a fraction of the population having the respective haplotype or genotype. For example, CY11ME is the mean of CYP2C9\*1\*1 activity in the population while CY11SD is the standard deviation and CY11FR is the relative frequency of the \*1\*1 genotype.

### 3.2.2 Genotype-related parameters

*Genetic variability distribution parameters*[4] ≡

```

{
```

*CYP2C9 population parameters*[5]

*VKOR population parameters*[6]

*Coagulation factor level parameters*[7]

*Vitamin K pool parameters*[8]

}

This macro is invoked in definition 26.

*CYP2C9 population parameters*[5]  $\equiv$

{

WCESMI: Float:= 0.00025;  
WCESMX: Float:= 0.125;  
WCERMI: Float:= 0.00025;  
WCERMX: Float:= 0.025;

CY11ME: Float:= 0.35;  
CY11SD: Float:= 0.02;

CY12ME: Float:= 0.28;  
CY12SD: Float:= 0.05;

CY13ME: Float:= 0.22;  
CY13SD: Float:= 0.1;

CY22ME: Float:= 0.17;  
CY22SD: Float:= 0.05;

CY23ME: Float:= 0.10;  
CY23SD: Float:= 0.05;

CY33ME: Float:= 0.1;  
CY33SD: Float:= 0.03;

CY1FR: Float:= 0.8;  
CY2FR: Float:= 0.15;  
CY3FR: Float:= 0.05;  
CY11FR: Float:= CY1FR \* CY1FR;  
CY12FR: Float:= CY1FR \* CY2FR \* 2.0;  
CY13FR: Float:= CY1FR \* CY3FR \* 2.0;  
CY22FR: Float:= CY2FR \* CY2FR;  
CY23FR: Float:= CY2FR \* CY3FR \* 2.0;  
CY33FR: Float:= CY3FR \* CY3FR;

}

This macro is invoked in definition 4.

*VKOR population parameters*[6]  $\equiv$

{

GCVMI: Float:= 0.0001;  
GCVMX: Float:= 0.0200;  
VKAFR: Float:= 0.43;  
VKGFR: Float:= 0.57;

VKGGFR: Float:= VKGFR \* VKGFR; -- Hardy-Weinberg  
VKGAFR: Float:= VKAFR \* VKGFR \* 2.0;  
VKA AFR: Float:= VKAFR \* VKAFR;

```
VKGGME: Float:= 0.34;
VKGGSD: Float:= 0.04;
```

```
VKGAME: Float:= 0.24;
VKGASD: Float:= 0.10;
```

```
VKAAME: Float:= 0.16;
VKAASD: Float:= 0.02;
```

```
}
```

This macro is invoked in definition 4.

```
Coagulation factor level parameters[7] ≡
{
```

```
  C7IME: Float:= 1.33 * 0.5;      -- data from [4] for normal
  C7ISD: Float:= 0.11 * 0.5;      -- renal donors
  C2IME: Float:= 1.33 * 0.5;
  C2ISD: Float:= 0.11 * 0.5;
  C10IME: Float:= 1.33 * 0.5;
  C10ISD: Float:= 0.11 * 0.5;
```

```
}
```

This macro is invoked in definition 4.

```
Vitamin K pool parameters[8] ≡
{
```

```
  FHPAR: Float:= 2.0;      -- initial amount of Vit-KH2
  CKME: Float:= 2.6;      -- total vitamin K
  CKSD: Float:= 0.1;
```

```
}
```

This macro is invoked in definition 4.

### 3.2.3 Distribution of anthropometric parameters and other parameters

Anthropometric features (weight and height) were taken from [5]. Height and weight are of course interdependent. The widely accepted definition of the body mass index,  $B = w/h^2$  ( $w$ : weight in kg,  $h$  height in m) was taken as describing the interdependency. We took the 'normal' height for weight given by body mass index for the averages above. Average B is 26.5 in males and 25.7 in females and there is of course a variation around 'average' weight for height. To generate reasonable heights and weights we take the  $h$  as an independent variable and generate  $w$  as  $Bh^2 + e$  where  $e$  is a normally distributed 'error (of  $w$  estimation from height)'. Lacking a reference, we estimated  $e$  using the following R function:

```
estimate BMI dispersion[9] ≡
{
```

```
  genwh <- function() {
    h <- rnorm(1000, 1.76, 0.069)
    w <- h * h * 26.55
    cat("males\n")
    cat(mean(w), "(", sd(w), ")")
```

```

hsd <- sd(w)
tgsd <- 12.41 # target dispersion :)
errsd <- sqrt(tgsd * tgsd - hsd * hsd)
w <- h * h * 26.5 + rnorm(1000, 0, errsd)
cat(mean(w), "(", sd(w), ") (with error, ", errsd, ")")

cat("females\n")
h <- rnorm(1000, 1.625, 0.0618)
w <- h * h * 25.7
cat(mean(w), "(", sd(w), ")")
hsd <- sd(w)
tgsd <- 12.42 # target dispersion :)
errsd <- sqrt(tgsd * tgsd - hsd * hsd)
w <- h * h * 25.7 + rnorm(1000, 0, errsd)
cat(mean(w), "(", sd(w), ") (with error, ", errsd, ")")
}
}

```

This macro is invoked in definition 27.

The population parameters used were:

*Anthropometric variability distribution parameters*[10]  $\equiv$

```

{
  WGMAME: Float:= 82.46;      -- weight, males
  WGMASD: Float:= 12.41;
  WGFEME: Float:= 67.88;    -- females
  WGFESD: Float:= 12.42;

  HTMAME: Float:= 1.7600;   -- height, males
  HTMASD: Float:= 0.0690;
  HTFEME: Float:= 1.6250;   -- females
  HTFESD: Float:= 0.0618;

  WGMAER: Float:= 10.74;
  WGFEEER: Float:= 11.23;

  -- avg bmis:

  BIMAME: Float:= WGMAME / HTMAME / HTMAME;
  BIFEME: Float:= WGFEME / HTFEME / HTFEME;

  GDISTF: Float:= 1.1;    -- distribution factor
  GWGPOW: Float:= 1.0;
}

```

This macro is invoked in definition 26.

### 3.3 Monte Carlo generation of individual constants

*Simulated case database*[11]  $\equiv$

```
{
```

```

Durmax: Natural := 504;
-- maximum duration of scenario in sampling intervals
Case_Max: Natural:= 100_000;
-- maximum number of cases in the database
Schedule_Max: Natural:= 100;
-- maximum number of drug doses given to a case

type Gender_Type is (Male, Female);

type Cyp_2c9_Genotype is ( CYP2C9_1_1, CYP2C9_1_2, CYP2C9_1_3,
                           CYP2C9_2_2, CYP2C9_2_3, CYP2C9_3_3);
type VKORC1_Genotype is ( VKORC1_A_A, VKORC1_G_A, VKORC1_G_G);

type Individual_Model is record

    CYP2C9: Cyp_2c9_Genotype := CYP2C9_1_1;
    VKORC1: VKORC1_Genotype := VKORC1_A_A;

    parameters of the system of equations[2]

    other individual phenotypic parameters[3]

    S_Clearance: Float:= 0.0;      -- to be computed
    R_Clearance: Float:= 0.0;      -- idem
end record;

type Administration is record
    T: Float:= 0.0;  -- hours from t_0
    D: Float:= 0.0;  -- mg
end record;

type Schedule is array(Integer range <>) of Administration;
subtype Schedule_Ix is Integer range 1..Schedule_Max;

type Inr_Trace is array (1..Durmax) of Float;

type Case_Model is record
    C: Individual_Model; -- that is, individual constants
    Q: Schedule(Schedule_Ix);
    Q_Fill: Schedule_Ix;
    Inr: Inr_Trace;
    F2,F7,F10,Fr,Fs,Fv,Fh: Inr_Trace;

    -- average dose for a target inr
    Dose_Found: Float:= 0.0; -- average daily dose to achieve INR
    Avg_Inr_Sup: Float:= 10000.0; -- INR for which dosage is seeked, immediately
above average
    Avg_Inr_Inf: Float:= 0.0; -- -- " -- immediately below average
    Dose_Sup, Dose_Inf: Float:= 0.0; -- simulated constant dosages above and below
interpolated dose_found
    Target_Inr: Float:= 2.5;
end record;

```

```
type Case_Database is array(Integer range <>) of Case_Model;
```

```
Db: Case_Database(1..Case_Max);  
Case_Fill: Integer range 0..Case_Max:= 0;
```

```
}
```

This macro is invoked in definition 26.

Monte Carlo generator of case parameters[12] ≡

```
{
```

```
is  
is  
procedure Generate_Random_Cases(Nc: Integer:= 100; Male_Proportion: Float:= 0.5)
```

```
Cygenotype: Float:= 0.0;  
Cymx, Cysd: Float:= 0.0;  
Vkgenotype: Float:= 0.0;  
Vkmx, Vksd: Float:= 0.0;
```

```
Cyfrs: array(Cyp_2C9_Genotype) of Float:=  
  (CY11FR, CY12FR, CY13FR, CY22FR, CY23FR, CY33FR);  
Cymes: array(Cyp_2C9_Genotype) of Float:=  
  (CY11ME, CY12ME, CY13ME, CY22ME, CY23ME, CY33ME);  
Cysds: array(Cyp_2C9_Genotype) of Float:=  
  (CY11SD, CY12SD, CY13SD, CY22SD, CY23SD, CY33SD);
```

```
Vkfrs: array(VKORC1_Genotype) of Float:= (VKA AFR, VKGA FR, VKGG FR);  
Vkmes: array(VKORC1_Genotype) of Float:= (VKA A ME, VKGA ME, VKGG ME);  
Vksds: array(VKORC1_Genotype) of Float:= (VKA ASD, VKGA SD, VKGG SD);
```

```
Cy: Cyp_2c9_Genotype:= CYP2C9_1_1;  
Vk: VKORC1_Genotype:= VKORC1_A_A;
```

```
begin
```

```
-- assert nc<case_max
```

```
Reset(Gn); -- a different scenario is generated with each run
```

```
-- set the random seed to obtain the same results with every run
```

```
for I in 1..Nc loop
```

```
  Cygenotype:= Random(Gn);
```

```
  Vkgenotype:= Random(Gn);
```

```
  for J in Cyp_2C9_Genotype loop
```

```
    Cygenotype := Cygenotype - Cyfrs(J);
```

```
    if Cygenotype <= 0.0 then
```

```
      Cy := J;
```

```
      Db(I).C.CYP2C9 := J;
```

```
      exit;
```

```
    end if;
```

```
  end loop;
```

```
  for J in VKORC1_Genotype loop
```

```
    Vkgenotype := Vkgenotype - Vkfrs(J);
```

```

        if Vkgenotype <= 0.0 then
            Vk := J;
            Db(I).C.VKORC1 := J;
            exit;
        end if;
    end loop;

    Db(I).C.Ces := WCESMI + (WCESMX - WCESMI) *
        Generate_Normal(Cymes(Cy), Cysds(Cy));
    if Db(I).C.Ces < WCESMI then
        Db(I).C.Ces := WCESMI;
    end if;
    if Db(I).C.Ces > WCESMX then
        Db(I).C.Ces := WCESMX;
    end if;

    Db(I).C.Cer := WCERMI + (WCERMX - WCERMI) *
        Generate_Normal(Cymes(Cy), Cysds(Cy));
    if Db(I).C.Cer < WCERMI then
        Db(I).C.Cer := WCERMI;
    end if;
    if Db(I).C.Cer > WCERMX then
        Db(I).C.Cer := WCERMX;
    end if;

    Db(I).C.Cv := GCVMI + (GCVMX - GCVMI) *
        Generate_Normal(Vkmes(Vk), Vksds(Vk));
    if Db(I).C.Cv < GCVMI then
        Db(I).C.Cv := GCVMI;
    end if;
    if Db(I).C.Cv > GCVMX then
        Db(I).C.Cv := GCVMX;
    end if;

    if Random(Gn) < Male_Proportion then
        Db(I).C.Gender := Male;
        Db(I).C.Height := Generate_Normal(HTMAME,HTMASD);
        Db(I).C.Weight := BIMAME * Db(I).C.Height * Db(I).C.Height +
            Generate_Normal(0.0,WGMAER);
    else
        Db(I).C.Gender := Female;
        Db(I).C.Height := Generate_Normal(HTFEME,HTFESD);
        Db(I).C.Weight := BIFEME * Db(I).C.Height * Db(I).C.Height +
            Generate_Normal(0.0,WGFEER);
    end if;

    Db(I).C.C7i := Generate_Normal(C7IME, C7ISD);
    Db(I).C.C2i := Generate_Normal(C2IME, C2ISD);
    Db(I).C.C10i := Generate_Normal(C10IME, C10ISD);
    Db(I).C.Ck := Generate_Normal(CKME, CKSD) * (0.6 + 0.4 *
    Db(I).C.Weight/80.0);
    end loop;
    Case_Fill := Nc;

```

```
end Generate_Random_Cases;
```

```
}
```

This macro is invoked in definition 26.

### 3.4 The PK/PD simulator

*INR trace simulator*[13]  $\equiv$

```
{
```

```
1) procedure Calculate_All_Trajectories(D: in out Case_Database; N_Tasks: Integer:=
```

```
task type Calculate_Trajectories(From,To: Integer);
```

```
type Calculate_Trajectories_Ref is access all Calculate_Trajectories;  
Cts: array(1..N_Tasks) of Calculate_Trajectories_Ref;
```

```
Lastn, Nextn: Integer:= 1;
```

```
task body Calculate_Trajectories is
```

```
Time_Scale: Float:= 1.0;
```

```
N_Variables: Integer:= 7;
```

```
Fr: constant Integer:= 1; -- F_R(t)
```

```
Fs: constant Integer:= 2;
```

```
Fv: constant Integer:= 3;
```

```
Fh: constant Integer:= 4;
```

```
F2: constant Integer:= 5;
```

```
F7: constant Integer:= 6;
```

```
F10: constant Integer:= 7;
```

```
Fin, Cer, Ces, Onev, Cv, Cs, Cr, Ck, Cd, Ci, Ca, C2, C2i, C7, C7i, C10,
```

```
C10i:
```

```
Float;
```

```
type Float_Type_array is array(integer range <>) of Float;
```

```
type Float_Type_array_ptr is access Float_Type_array;
```

```
type double_Float_Type_array is array(integer range <>,integer range <>)  
of Long_Float;
```

```
type double_Float_Type_array_ptr is access double_Float_Type_array;
```

```
procedure derivs(x:in float;y:in float_type_array;z:out float_type_array) is  
begin
```

```
Z(Fr) := Time_Scale * (Fin - Cer * Y(Fr));
```

```
Z(Fs) := Time_Scale * (Fin - Ces * Y(Fs));
```

```
Z(Fv) := Time_Scale * (Cv * Onev - Cv * Y(Fv)  
- Cs * Y(Fs) * Y(Fv)  
- Cr * Y(Fr) * Y(Fv));
```

```
Z(Fh) := Time_Scale * ((Ck - Y(Fh))*(Cd*Y(Fv) + Ci) - Ca*Y(Fh));
```

```
Z(F2) := Time_Scale * ( C2 * C2i * Y(Fh) - C2 * Y(F2));
```

```

    Z(F7) := Time_Scale * ( C7 * C7i * Y(Fh) - C7 * Y(F7));
    Z(F10) := Time_Scale * ( C10 * C10i * Y(Fh) - C10 * Y(F10));
end;

subtype Farray is Float_Type_Array(1..N_Variables);

eps:float:=0.000001;
first_step:float:=0.001;
ystart: Farray; -- float_type_array(1..n_variables);
Eulery, Eulerz: Farray;
Xpmin: Float:= 10000000000.0;

procedure Calculate_With_Euler is

    Euler_Step: Float:= 0.03;
    N_Euler_Steps: Integer;

begin

    Time_Scale := Euler_Step;

    for Di in From..To loop
        Ystart(Fr) := 0.0;
        Ystart(Fs) := 0.0;
        Ystart(Fv) := 1.0;
        Ystart(Fh) := FHPAR;
        Ystart(F2) := 2.0 * D(Di).C.C2i;
        Ystart(F7) := 2.0 * D(Di).C.C7i;
        Ystart(F10) := 2.0 * D(Di).C.C10i;

        Fin := 0.0;
        Cer := D(Di).C.Cer;
        Ces := D(Di).C.Ces;
        Onev := D(Di).C.Onev;
        Cv := D(Di).C.Cv;
        Cs := D(Di).C.Cs;
        Cr := D(Di).C.Cr;
        Ck := D(Di).C.Ck;
        Cd := D(Di).C.Cd;
        Ci := D(Di).C.Ci;
        Ca := D(Di).C.Ca;
        C2 := D(Di).C.C2;
        C2i := D(Di).C.C2i;
        C7 := D(Di).C.C7;
        C7i := D(Di).C.C7i;
        C10 := D(Di).C.C10;
        C10i := D(Di).C.C10i;

        for I in 1..Durmax loop -- hours
            Fin := 0.0;
            for J in Schedule_Ix loop
                if abs(D(Di).Q(J).T - Float(I)) <= 0.5 then
                    Fin := D(Di).Q(J).D;
                end if;
            end for;
        end for;
    end for;
end procedure;

```

```

        end if;
    end loop;

    N_Euler_Steps := Integer(1.0 / Euler_Step);

    Eulery := Ystart;

    for J in 1..N_Euler_Steps loop
        Derivs(0.0, Eulery, Eulerz);
        for K in 1..N_Variables loop
            Eulery(K) := Eulery(K) + Eulerz(K);
        end loop;
    end loop;

    D(Di).F2(I) := Eulery(F2);
    D(Di).F7(I) := Eulery(F7);
    D(Di).F10(I) := Eulery(F10);
    D(Di).Fv(I) := Eulery(Fv);
    D(Di).Fr(I) := Eulery(Fr);
    D(Di).Fs(I) := Eulery(Fs);
    D(Di).Fh(I) := Eulery(Fh);
    D(Di).Inr(I) := 14.72313/(((Eulery(F7)*100.0)**0.296) *
                                ((Eulery(F2)*100.0)**0.251));

        Ystart := Eulery;
    end loop;
end loop;

end Calculate_With_Euler;

begin
    Calculate_With_Euler;
end Calculate_Trajectories;

begin
    for I in 1..N_Tasks loop
        if Lastn > Case_Fill then
            exit;
        end if;
        Nextn := (Case_Fill * I) / N_Tasks;
        Cts(I) := new Calculate_Trajectories(Lastn, Nextn);
        Lastn := Nextn+1;
    end loop;
end Calculate_All_Trajectories;

}

```

This macro is invoked in definition 26.

## 4 Simulation result functionals and calibration

### 4.1 Functionals on the case space

Functionals on the case space<sup>[14]</sup>  $\equiv$

```
{  
  
function S_Warfarin_Clearance(Icase: Integer:= 1; From: Integer:= 48) return Float  
is  
  
begin  
  --      Put_Line(Float'Image(Db(Icase).Fr(From)) & " - "  
  --      & Float'Image(Db(Icase).Fr(From+1)));  
  return ((Db(Icase).Fs(From) - Db(Icase).Fs(From+1)) / Db(Icase).Fs(From))  
    * Db(Icase).C.Blood_Volume * 1000.0 / 60.0 / Db(Icase).C.Weight;  
  -- ml / min / kg  
end S_Warfarin_Clearance;  
  
function Average_Inr(Icase: Integer:= 1;  
                    From: Integer:= Durmax-24;  
                    To: Integer:= Durmax-1) return Float is  
  
  Ainr: Float:= 0.0;  
  
begin  
  for I in From..To loop  
    Ainr := Ainr + Db(Icase).Inr(I);  
  end loop;  
  return Ainr / Float(To - From + 1);  
end Average_Inr;  
  
function Simple_Raise_Time(Icase: Integer:= 1;  
                          Raise_Ratio: Float:= 0.95;  
                          From: Integer:= 0;  
                          To: Integer:= Durmax-1) return Float is  
  
  Inrmax: Float;  
  
begin  
  Inrmax := Average_Inr(Icase,To-23,To);  
  for I in From..To loop  
    if Db(Icase).Inr(I) > 1.0 + (Raise_Ratio * (Inrmax - 1.0)) then  
      return Float(I-From);  
    end if;  
  end loop;  
  raise Internal_Inconsistency;  
end Simple_Raise_Time;  
  
}
```

This macro is invoked in definition 26.

## 4.2 Calibration experiments

Parameters were adjusted so that the dynamic INR response found in [6, figure 1] and the gender and genotype dependence of the maintenance dosage of warfarin reported in [2, figure 2] would be reproduced when similar samples of cases are drawn from our randomly generated population.

Before these calibrations, a number of simpler checks and adjustments have been made to get the set of parameters in a reasonable range, that are called 'basic checks' below.

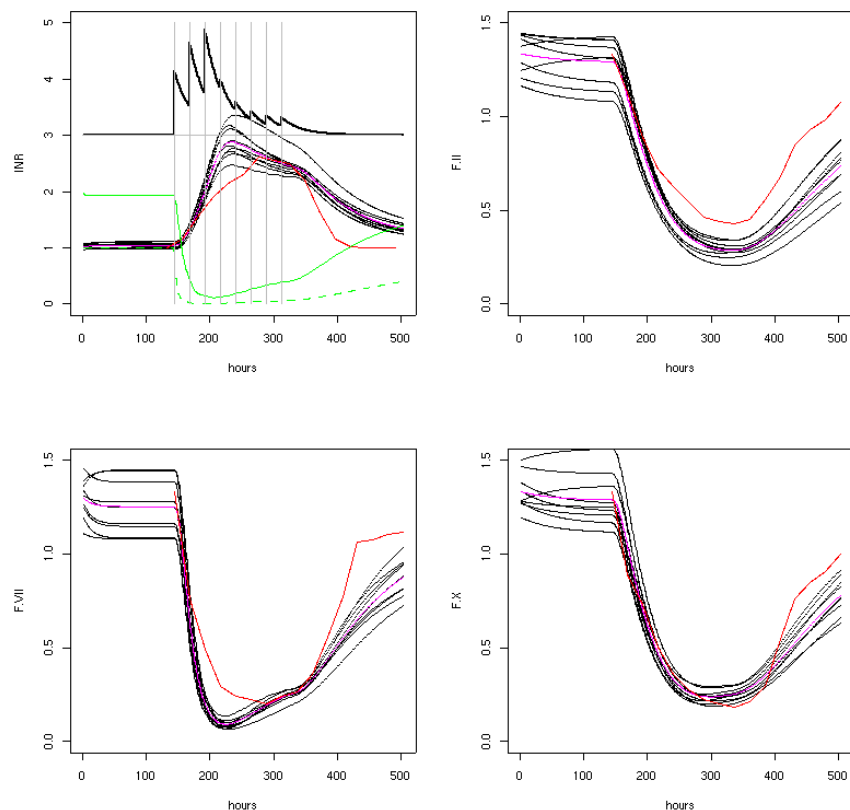


Figure 4. Calibration results attempting to reproduce from [6, figure 1] by randomly generating 6 males and 3 females from the general population. *Top left.* Thick black upper trace: S-warfarin blood concentration in one of the cases. Vertical gray bars: instants of warfarin administration; the first three peaks correspond to the 'loading' dose of 10mg warfarin per day while the next to the five daily doses adjusted for the individual. The red line is the experimental instantaneous INR average from [6]. The black lines are individual INR(t) graphics in the simulated cases. The magenta line is the average of the simulated cases. The continuous green line is the ammount of reduced vitamin-K in one of the cases. The dotted green line is the activity of the simulated VKORC1 in one of the cases. These two graphics should be compared with [7, figures 5,6]. *Other graphics.* With the same conventions as in the top left they represent the time course of factors II, VII and X. Factor X was not further taken into account for INR calculation in this version.

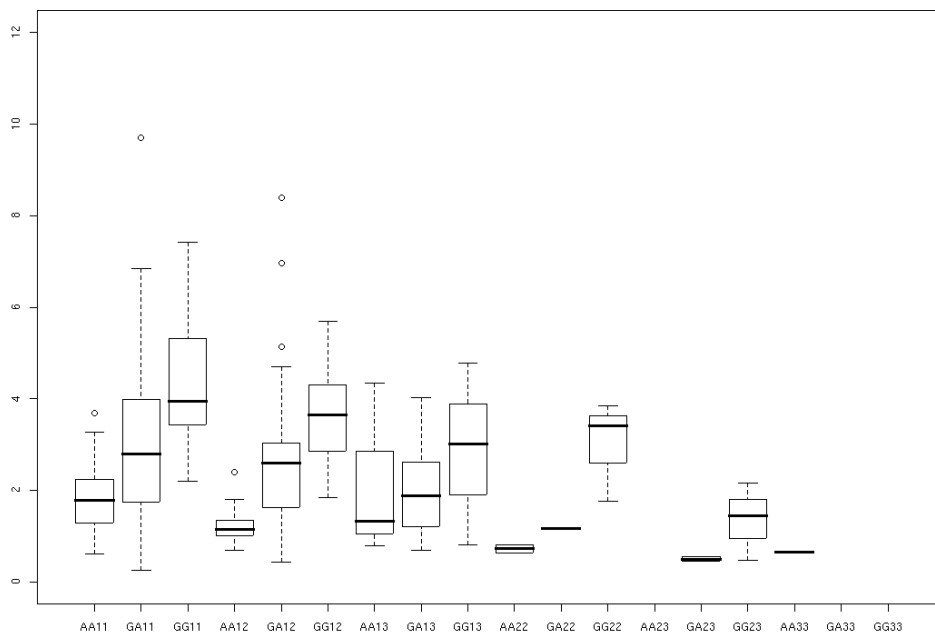


Figure 5. Calibration results trying to reproduce [2, figure 1]. On  $x$ —individual genotype combinations of the AG locus on the VKORC1 gene and the \*1,\*2,\*3 haplotype combinations for CYP2C9. On  $y$ —the average daily warfarin doses for 289 cases generated as from the general european population. Boxes represent medians, first and third quartiles for each genetic subset.

```

Calibration experiments[15] ≡
{
    Basic checks[16]
    Calibration[18]
}

```

This macro is invoked in definition 26.

### 4.2.1 Basic checks

Check that the range of expected baseline INR values is consistent with the ranges of VKORC1 and CYP2C9 activity.

```

Basic checks[16] ≡
{
    procedure Generate_Uniform_Wes_Wv(Ces_Min:      Float:= WCESMI;
                                       Ces_Max:      Float:= WCESMX;
                                       Cer_Min:      Float:= WCERMI;
                                       Cer_Max:      Float:= WCERMX;
                                       Cv_Min:       Float:= GCVMI;
                                       Cv_Max:       Float:= GCVMX;

```

Wes\_N,Wv\_N: Integer:= 10) is

```
I: Integer;

begin
  if Wes_N * Wv_N > Case_Max then
    raise Internal_Inconsistency;
  end if;
  for Wes_I in 1..Wes_N loop
    for Wv_I in 1..Wv_N loop
      I := Wv_I + (Wes_I - 1) * Wv_N;
      if Wes_N = 1 then
        Db(I).C.Ces := Ces_Min;
        Db(I).C.Cer := Cer_Min;
      else
        Db(I).C.Ces := Ces_Min +
          Float(Wes_I-1) * (Ces_Max - Ces_Min) / Float(Wes_N-1);
        Db(I).C.Cer := Cer_Min +
          Float(Wes_I-1) * (Cer_Max - Cer_Min) / Float(Wes_N-1);
      end if;
      if Wv_N = 1 then
        Db(I).C.Cv := Cv_Min;
      else
        Db(I).C.Cv := Cv_Min +
          Float(Wv_I-1) * (Cv_Max - Cv_Min) / Float(Wv_N-1);
      end if;
    end loop;
  end loop;
  Case_Fill := Wv_N * Wes_N;
end Generate_Uniform_Wes_Wv;

procedure Calculate_And_Print_Baseline_Inr(N_Wes,N_Wv: Integer) is

  Crto: File_Type;

begin
  Generate_Uniform_Wes_Wv(Wes_N => N_Wes, Wv_N => N_Wv);
  Generate_Constant_Daily_Dose(0.0, From_Day => 1.8);
  Calculate_All_Trajectories(Db,2);

  Create(Crto, Name => "baseline_inr.dat");
  for Wes in 1..N_Wes loop
    for Wv in 1..N_Wv loop
      Put(Crto,Float'Image(Average_Inr(Wv + N_Wv * (N_Wes-1))));
    end loop;
    New_Line(Crto);
  end loop;
  Close(Crto);
end Calculate_And_Print_Baseline_Inr;

procedure Average_Inr_By_Wes_Wv is

begin
```

```

Generate_Uniform_Wes_Wv(Wes_N => 8, Wv_N => 8);
Generate_Constant_Daily_Dose(5.0, From_Day => 1.8);
Calculate_All_Trajectories(Db,2);
for I in 1..Case_fill loop
  Put(Float'Image(Average_Inr(I)));
  if I mod 8 = 0 then
    New_Line;
  end if;
end loop;
end Average_Inr_By_Wes_Wv;

procedure Raise_Time_By_Wes_Wv(Warf_Dose: Float:= 5.0) is
begin
  Generate_Uniform_Wes_Wv(Wes_N => 8, Wv_N => 8);
  Generate_Constant_Daily_Dose(Warf_Dose, From_Day => 1.8);
  Calculate_All_Trajectories(Db,2);
  for I in 1..Case_fill loop
    Put(Integer'Image(Integer(Simple_Raise_Time(I, From => 44))));
    if I mod 8 = 0 then
      New_Line;
    end if;
  end loop;
end Raise_Time_By_Wes_Wv;

}

```

This macro is invoked in definition 15.

## 4.2.2 Calibration procedures

**Find\_Doses\_For\_Target** calculates the average individual daily dosage that is needed to reach an ideal average INR after the treatment response stabilises, given the case parameters. The average dose for reaching the common target of INR=2.5 and the associated individual parameters in each case are saved in a file with procedure **Save\_Db** and then represented graphically and summarized with the R functions **f1.sconce.blood2005** and **s.dose.assoc.sconce2005**. They result in figure 5, that should be compared with [2, figure 1] and table 1 that compares a variety of statistics on average daily variability dose from a simulated population with those in [2], section ‘associations with warfarin dose’.

---

subset	simulated	measured [2]
(gender)		
females	2.478	2.9
males	3.001	3.7
(CYP2C9)		
*1*1	3.367 (1.623)	4.08 (2.13)
*1*2	2.548 (1.023)	3.56 (1.82)
*1*3	2.721 (1.887)	2.70 (1.38)
*2*2	1.975 (0.4802)	1.92 (1.12)
*2/3*3	1.453 (0.2986)	1.58 (0.79)
(VKORC1)		
GG	3.78 (1.599)	4.53 (2.22)
GA	2.834 (1.502)	3.83 (1.91)
AA	2.058 (0.6583)	2.23 (1.26)

---

Table 1. Medians of daily average warfarin dose for maintaining an INR of 2.5 in males and females; sample means and standard deviations of average warfarin by genotype in one simulated sample and an actual population sample reported in [2].

The calibration output is summarized in file `expereport.html` that can be obtained by calling the R function `exper.rep.report`.

*Calculate ideal individual dosage*[17]  $\equiv$

```
{
  procedure Find_Doses_For_Target(Target_Inr: Float:= 2.5;
    Tolerance: Float:= 0.25) is
    Avg: Float;
    Verbosef: Boolean:= False;
  begin
    for I in 1..Case_Fill loop
      Db(I).Avg_Inr_Sup := 10.0;
      Db(I).Avg_Inr_Inf := 0.0;
      Db(I).Dose_Sup := 40.0;
      Db(I).Dose_Inf := 0.0;
    end loop;
    for D in 1..80 loop
      if Verbosef then
        Put_Line("doing dose " & Integer'Image(D));
```

```

end if;
Generate_Constant_Daily_Dose(Float(D)/2.0);
Calculate_All_Trajectories(Db, N_Cores);
for I in 1..Case_Fill loop
  Avg := Average_Inr(I);
  if Avg >= Target_Inr and then Avg < Db(I).Avg_Inr_Sup then
    Db(I).Avg_Inr_Sup := Avg;
    Db(I).Dose_Sup := Float(D)/2.0;
  end if;
  if Avg <= Target_Inr and then Avg > Db(I).Avg_Inr_Inf then
    Db(I).Avg_Inr_Inf := Avg;
    Db(I).Dose_Inf := Float(D)/2.0;
  end if;
end loop;
end loop;

for I in 1..Case_Fill loop
  Db(I).Dose_Found := Db(I).Dose_Inf +
    (Target_Inr - Db(I).Avg_Inr_Inf) * (Db(I).Dose_Sup - Db(I).Dose_Inf) /
    (Db(I).Avg_Inr_Sup - Db(I).Avg_Inr_Inf);
  if Verbosef then
    Put_Line(Float'Image(Db(I).Avg_Inr_Sup) & " " &
      Float'Image(Db(I).Avg_Inr_Inf) & " " &
      Float'Image(Db(I).Dose_Sup) & " " &
      Float'Image(Db(I).Dose_Inf) & " " &
      Float'Image(Db(I).Dose_Found));
  end if;
end loop;

end Find_Doses_For_Target;

}

```

This macro is invoked in definition 18.

The `Dangelo_2002_Experiment` procedure saves the coagulation factor levels and the INR for a small sample of simulated individuals as in the experiment reported in [6]. These results are then represented graphically by the separate R function `dangelo2002.inr.graph`.

*Calibration*[18]  $\equiv$   
{

*Calculate ideal individual dosage*[17]

procedure `Dangelo_2002_Experiment`(N\_Cases: Integer:= 9) is

begin

  Generate\_Random\_Cases(N\_Cases, 6.0/9.0);

  Find\_Doses\_For\_Target(2.2);

  for I in 1..Case\_Fill loop

    for J in 1..Schedule\_Max loop

      Db(I).Q(J).T := 0.0;

      Db(I).Q(J).D := 0.0;

    end loop;

  for J in 1..8 loop

```

        Db(I).Q(J).T := 24.0 * (Float(J) + 5.0);
    end loop;
    for J in 1..3 loop
        Db(I).Q(J).D := 10.0 / (GDISTF * Db(I).C.Weight ** GWGPOW);
    end loop;
    for J in 4..8 loop
        Db(I).Q(J).D := Db(I).Dose_Found / (GDISTF * Db(I).C.Weight ** GWGPOW);
    end loop;
    Db(I).Q_Fill := 8;
end loop;
Calculate_All_Trajectories(Db,N_Cores);
Save_Signals(Inr => True, Inr_Name => "dangelo2000_inr.dat",
            F2 => True, F2_Name => "dangelo2000_f2.dat",
            F7 => True, F7_Name => "dangelo2000_f7.dat",
            F10 => True, F10_Name => "dangelo2000_f10.dat",
            Fh => True, Fh_Name => "dangelo2000_fh.dat",
            Fv => True, Fv_Name => "dangelo2000_fv.dat",
            Fs => True, Fs_Name => "dangelo2000_fs.dat");
for I in 1..N_Cases loop
    Db(I).C.S_Clearance:= S_Warfarin_Clearance(I,193);
end loop;
Save_Db("dangelo2000_db.dat");
end Dangelo_2002_Experiment;
}

```

This macro is invoked in definition 15.

## 5 Bayesian experiment

### 5.1 Variables

*Bayesian experiment variables*[19]  $\equiv$

{

Wv\_N, Wes\_N: Integer:= 50;

Max\_N\_Inr: constant Natural:= 20; -- maximum nr of simulated measurements

Inr\_Min: constant Float := 0.0;

Inr\_Max: constant Float := 10.0;

Inr\_N\_Bins: constant Natural := 100; -- for the whole range, 0.1 per bin

Inr\_Resolution: constant Float := (Inr\_Max - Inr\_Min) / Float(Inr\_N\_Bins); -- 0.1

Inrp: array(1..Inr\_N\_Bins, 1..Durmax) of Float :=  
 (others => (others => 0.0));

Probdb: array(1..Wes\_N, 1..Wv\_N, 0..Max\_N\_Inr) of Float :=  
 (others => (others => (others => 0.0)));

Pii: array(1..Case\_Max) of Float;

```

Pii_Db: array(1..Case_Max, 0..Max_N_Inr) of Float;
Wes_I, Wv_I: array(1..Case_Max) of Integer;
Dw, Dwes, Dwv: Float;

Crti: Natural:= 0;
Inr_Bin: Natural;

DataI: array(0..Max_N_Inr) of Natural := (others => 1);
Datao: array(0..Max_N_Inr) of Float; -- original, simulated INR
Datam: array(0..Max_N_Inr) of Float; -- measured INR (original + error)
Datap: array(0..Max_N_Inr) of Float; -- mean predicted INR
DataSt: array(0..Max_N_Inr) of Float; -- stability of predicted INR
Entro: array(0..Max_N_Inr) of Float;
Rentro: array(0..Max_N_Inr) of Float;
Rentro2: array(0..Max_N_Inr) of Float;

Measure_N: Natural := 20;
Simulated: Integer:= 17; -- index of the simulated case, 1..case_fill
Measure_First: Natural := 150;
Measure_Every: Natural := 15;
Relative_Inr_Error: Float := 0.1;

Prel2info, Prelinfo, Pinfo, Pp, Ppp, Mean_Predicted_Inr, Sd_Predicted_Inr: Float;
N_Measure_Sets: Integer:= 50;
}

```

This macro is invoked in definition 26.

## 5.2 Procedure

*Bayesian experiment procedure*[20]  $\equiv$   
{

```

procedure Bayes_Correction_Experiment_MC is

  Crto, Crtmo, Crtinr: File_Type;

  K,L: Integer;

  Full_Disclosure: Boolean:= True;

  function Nmi(M: Integer) return String is

    Toi: String:= Integer'Image(M);

  begin
    return Toi(2..Toi'Last);
  end Nmi;

begin
  Generate_Random_Cases(Nc => Case_Max); -- number of cases to use
  Generate_Constant_Daily_Dose(5.0,1.8);

```

```

Calculate_All_Trajectories(Db,2);

if Full_Disclosure then
  Save_Db("bayexpmc.dat");
  Save_Signals(Inr => True, Inr_Name => "bayexpmc_inr.dat");
end if;

Create(Crtmo, Name => "bayexpmc_inr_series.dat");
Put_Line(Crtmo, "MSET SIM T INR INRO INRP INRST ENTROPY RENTROPY RENTROPY2");

-- generate measurements

for Mset in 1..N_Measure_Sets loop

  Simulated := 1 + Integer(Random(Gn)*Float(Case_Fill-1));

  for I in 1..Measure_N loop
    Datai(I) := Measure_First + Measure_Every * (I-1);
    Datam(I) := Generate_Normal(Mu => Db(Simulated).Inr(Datai(I)),
                               Sigma => Db(Simulated).Inr(Datai(I)) *
                               Relative_Inr_Error);
    Datao(I) := Db(Simulated).Inr(Datai(I));
  end loop;
  -- calculation of the a priori density

  Dw := 1.0/Float(Case_Fill);

  Dwes := WCESMX / Float(Wes_N) / 2.0;
  Dwv := GCVMX / Float(Wv_N) / 2.0;

  for I in 1..Wes_N loop
    for J in 1..Wv_N loop
      for K in 0..Measure_N loop
        Probdb(I,J,K) := 0.0;
      end loop;
    end loop;
  end loop;

  Inrp := (others => (others => 0.0));

  for I in 1..Case_Fill loop
    Pii(I) := Dw;
    K := 1 + Integer(Db(I).C.Ces / Dwes);
    L := 1 + Integer(Db(I).C.Cv / Dwv);
    if K>Wes_N then
      K := Wes_N;
    end if;
    if L>Wv_N then
      L := Wv_N;
    end if;
    Wes_I(I) := K;
    Wv_I(I) := L;
    Probdb(K,L,0) := Probdb(K,L,0) + Pii(I);
  end loop;
end loop;

```

```

    Pii_Db(I,0) := Pii(I);
end loop;

-- output of a priori probabilities of INR(t) if no
-- simulated measurements are used

if Generate_I then
    Create(Crto, Name => "apriori_inr_limits.dat");
    Crti := 0;
    for T in 1..Durmax loop
        for J in 1..Case_Fill loop
            Inr_Bin := Integer(Db(J).Inr(T) / Inr_Resolution) + 1;
            if Inr_Bin > Inr_N_Bins then
                Inr_Bin := Inr_N_Bins;
            end if;
            Inrp(Inr_Bin,T) := Inrp(Inr_Bin,T) + Pii_Db(J,0);
        end loop;
        Ppp := 0.0;
        for I in 1..Inr_N_Bins loop
            Ppp := Ppp + Inrp(I,T);
        end loop;
        for I in 1..Inr_N_Bins loop
            Inrp(I,T) := Inrp(I,T) / Ppp;
            Put(Crto, Float'Image(Inrp(I,T)));
        end loop;
        New_Line(Crto);
    end loop;
    Close(Crto);
end if;

-- calculation of a posteriori probabilities

if Generate_P then
    Create(Crto, Name => "bayexpmc_postprob.dat" & Nmi(Mset));
    Create(Crtinr, Name => "inr_" & Nmi(Simulated));
    for I in 1..Durmax loop
        Put_Line(Crtinr, Float'Image(Db(Simulated).Inr(I)));
    end loop;
    Close(Crtinr);
end if;
for I in 1..Measure_N loop
    Ppp := 0.0;
    Pinfo := 0.0;
    Prelinfo := 0.0;
    Prel2info := 0.0;
    Mean_Predicted_Inr := 0.0;
    Sd_Predicted_Inr := 0.0;
    for J in 1..Case_Fill loop
        Pp := Normal_Density(Datam(I), Mu => Db(J).Inr(Datai(I)),
                             Sigma => Db(J).Inr(Datai(I))/10.0);
        Pii(J) := Pii(J) * Pp;
        Ppp := Ppp + Pii(J);
    end loop;
end loop;

```

```

end loop;
for J in 1..Case_Fill loop
  Pii(J) := Pii(J) / Ppp;
  Pii_Db(J,I) := Pii(J);
  Probdb(Wes_I(J),Wv_I(J),I) := Probdb(Wes_I(J),Wv_I(J),I) + Pii(J);
end loop;

for Wes in 1..Wes_N loop
  for Wv in 1..Wv_N loop
    if Probdb(Wes,Wv,I)>0.0 and then Probdb(Wes,Wv,I-1)>0.0 then
      Pinfo := Pinfo - Probdb(Wes,Wv,I) * Log(Probdb(Wes,Wv,I),2.0);
      Prelinfo := Prelinfo + Probdb(Wes,Wv,I) *
        Log(Probdb(Wes,Wv,I)/Probdb(Wes,Wv,I-1),2.0);
      Prel2info := Prel2info + Probdb(Wes,Wv,I) *
        Log(Probdb(Wes,Wv,I)/Probdb(Wes,Wv,0),2.0);
    end if;
    if Generate_P then
      Put(Crto, Float'Image(Probdb(Wes,Wv,I-1)));
    end if;
  end loop;
end loop;

for J in 1..Case_Fill loop
  Mean_Predicted_Inr := Mean_Predicted_Inr + Db(J).Inr(Datai(I)) *
Pii(J);
end loop;
for j in 1..Case_Fill loop
  Sd_Predicted_Inr := Sd_Predicted_Inr + ((Db(J).Inr(Datai(I)) -
Mean_Predicted_Inr) *
  (Db(J).Inr(Datai(I)) - Mean_Predicted_Inr) * Pii(J));
end loop;
Entro(I) := Pinfo;
Rentro(I) := Prelinfo;
Rentro2(I) := Prel2info;
Datap(I) := Mean_Predicted_Inr;
Datast(I) := Mean_Predicted_Inr / Sqrt(Sd_Predicted_Inr) ;
if Generate_P then
  New_Line(Crto);
end if;
end loop;
if Generate_P then
  Close(Crto);
end if;

-- calculation of confidence limits and probability
-- distribution of the INR at every instant

if Generate_I then
  Inrp := (others => (others => 0.0));
  Create(Crto, Name => "bayexpmc_inr_limits.dat" & Nmi(Mset));
  Crti := 0;
  for T in 1..Durmax loop
    while Crti < Measure_N and then T>=Datai(Crti+1) loop

```

```

        Crti := Crti + 1;
    end loop;
    for J in 1..Case_Fill loop
        Inr_Bin := Integer(Db(J).Inr(T) / Inr_Resolution) + 1;
        if Inr_Bin > Inr_N_Bins then
            Inr_Bin := Inr_N_Bins;
        end if;
        Inrp(Inr_Bin,T) := Inrp(Inr_Bin,T) + Pii_Db(J,Crti);
    end loop;
    Ppp := 0.0;
    for I in 1..Inr_N_Bins loop
        Ppp := Ppp + Inrp(I,T);
    end loop;
    for I in 1..Inr_N_Bins loop
        Inrp(I,T) := Inrp(I,T) / Ppp;
        Put(Crto, Float'Image(Inrp(I,T)));
    end loop;
    New_Line(Crto);
end loop;
Close(Crto);
end if;

-- output of simulated INR measurements and entropies
-- (entropies not discussed here)

for I in 1..Measure_N loop
    Put_Line(Crtmo, Integer'Image(Mset) & Integer'Image(Simulated) &
        Integer'Image(Datai(I)) & Float'Image(Datam(I)) &
        Float'Image(Datao(I)) & Float'Image(Datap(I)) &
Float'Image(Datast(I))
        & Float'Image(Entro(I))
        & Float'Image(Rentro(I))& Float'Image(Rentro2(I)));
    end loop;
end loop;
Close(Crtmo);
end Bayes_Correction_Experiment_MC;

}

```

This macro is invoked in definition 26.

## 6 Output

### 6.1 Output switch variables

*Output switch variables*[21] ≡  
{

```

Generate_Db: Boolean:= True;
Generate_P: Boolean:= True;

```

```

Generate_I: Boolean:= True;
Generate_M: Boolean:= True;

```

```

}

```

This macro is invoked in definition 26.

## 6.2 General output procedures

*General output procedures*[22] ≡

```

{
  procedure Save_Db(Filename: String:= "pwissdb.dat") is

    Co: File_Type;

  begin
    Create(Co, Name => Filename);
    Put_Line(Co, "n sex CYP2C9 VKORC1 cer ces cv ck c7i c2i weight height
avgdose sclr rclr");
    for I in 1..Case_Fill loop
      Put(Co, Integer'Image(I));
      Put(Co, " " & Gender_Type'Image(Db(I).C.Gender)(1) & "");
      Put(Co, " " & CYP_2C9_Genotype'Image(Db(I).C.CYP2C9) & "");
      Put(Co, " " & VKORC1_Genotype'Image(Db(I).C.VKORC1) & "");
      Put(Co, " " & Float'Image(Db(I).C.Cer));
      Put(Co, " " & Float'Image(Db(I).C.Ces));
      Put(Co, " " & Float'Image(Db(I).C.Cv));
      Put(Co, " " & Float'Image(Db(I).C.Ck));
      Put(Co, " " & Float'Image(Db(I).C.C7i));
      Put(Co, " " & Float'Image(Db(I).C.C2i));
      Put(Co, " " & Float'Image(Db(I).C.Weight));
      Put(Co, " " & Float'Image(Db(I).C.Height));
      if Db(I).Dose_Found=0.0 then
        Put(Co, " NA");
      else
        Put(Co, " " & Float'Image(Db(I).Dose_Found));
      end if;
      if Db(I).C.S_Clearance=0.0 then
        Put(Co, " NA");
      else
        Put(Co, " " & Float'Image(Db(I).C.S_Clearance));
      end if;
      if Db(I).C.R_Clearance=0.0 then
        Put(Co, " NA");
      else
        Put(Co, " " & Float'Image(Db(I).C.R_Clearance));
      end if;
      New_Line(Co);
    end loop;
    Close(Co);
  end Save_Db;

  procedure Save_Signals(Inr, F7, F10, F2, Fr, Fs, Fv, Fh: Boolean:= False;

```

```

        Inr_Name: String:= "pwissinr.dat";
        F7_Name: String:= "pwissf7.dat";
        F10_Name: String:= "pwissf10.dat";
        F2_Name: String:= "pwissf2.dat";
        Fr_Name: String:= "pwissfr.dat";
        Fs_Name: String:= "pwissfs.dat";
        Fv_Name: String:= "pwissfv.dat";
        Fh_Name: String:= "pwissfh.dat"
    ) is

```

```

Co: File_Type;

```

```

begin
    if Inr then
        Create(Co, Name => Inr_Name);
        for I in 1..Case_Fill loop
            for J in 1..Durmax loop
                Put(Co, Float'Image(Db(I).Inr(J)) & " ");
            end loop;
            New_line(Co);
        end loop;
    end if;
    Close(Co);
    if F7 then
        Create(Co, Name => F7_Name);
        for I in 1..Case_Fill loop
            for J in 1..Durmax loop
                Put(Co, Float'Image(Db(I).F7(J)) & " ");
            end loop;
            New_line(Co);
        end loop;
        Close(Co);
    end if;
    if F2 then
        Create(Co, Name => F2_Name);
        for I in 1..Case_Fill loop
            for J in 1..Durmax loop
                Put(Co, Float'Image(Db(I).F2(J)) & " ");
            end loop;
            New_line(Co);
        end loop;
        Close(Co);
    end if;
    if F10 then
        Create(Co, Name => F10_Name);
        for I in 1..Case_Fill loop
            for J in 1..Durmax loop
                Put(Co, Float'Image(Db(I).F10(J)) & " ");
            end loop;
            New_line(Co);
        end loop;
        Close(Co);
    end if;
end

```

```

if Fr then
  Create(Co, Name => Fr_Name);
  for I in 1..Case_Fill loop
    for J in 1..Durmax loop
      Put(Co, Float'Image(Db(I).Fr(J)) & " ");
    end loop;
    New_line(Co);
  end loop;
  Close(Co);
end if;
if Fs then
  Create(Co, Name => Fs_Name);
  for I in 1..Case_Fill loop
    for J in 1..Durmax loop
      Put(Co, Float'Image(Db(I).Fs(J)) & " ");
    end loop;
    New_line(Co);
  end loop;
  Close(Co);
end if;
if Fv then
  Create(Co, Name => Fv_Name);
  for I in 1..Case_Fill loop
    for J in 1..Durmax loop
      Put(Co, Float'Image(Db(I).Fv(J)) & " ");
    end loop;
    New_line(Co);
  end loop;
  Close(Co);
end if;
if Fh then
  Create(Co, Name => Fh_Name);
  for I in 1..Case_Fill loop
    for J in 1..Durmax loop
      Put(Co, Float'Image(Db(I).Fh(J)) & " ");
    end loop;
    New_line(Co);
  end loop;
  Close(Co);
end if;
end Save_Signals;

```

}

This macro is invoked in definition 26.

## 7 System integration

### 7.1 System options

*System variables*[23]  $\equiv$

```
{  
  
    N_Cores: Positive:= 1;  
    Use_Runge_Kutta: Boolean:= False;  
    N_Processors: Integer:= 1;  
  
}
```

This macro is invoked in definition 26.

### 7.2 Utility function and procedures

*Utility functions*[24]  $\equiv$

```
{  
  
    -- utility functions  
  
    function Normal_Density(X,Mu,Sigma: Float) return Float is  
  
    begin  
        return 1.0/(Sqrt(2.0 * Pi) * Sigma) * Exp(-((X - Mu)**2/(2.0*(Sigma**2))));  
    end Normal_Density;  
  
    Norm_Cumul: array(-1000..1000) of Float; -- -10 to +10 sigma  
    Inv_Norm_Cumul: array(0..1000) of Float; -- nr of sd from 0 for  
                                                -- cumul probability 1/1000  
  
    procedure Calculate_Norm_Cumul is -- gaussian cumulative distribution  
                                        -- for generating normally distributed random  
                                        -- numbers with average 0, dispersion 100  
  
    begin  
        Norm_Cumul(-1000) := 0.0;  
        for I in -999..1000 loop  
            Norm_Cumul(I) := Norm_Cumul(I-1) + Normal_Density(Float(I),0.0,100.0);  
            -- if I mod 100 = 0 then  
            -- Put_Line(Integer'Image(I) & Float'Image(Norm_Cumul(I)));  
            -- end if;  
        end loop;  
  
        for I in -500..500 loop  
            Inv_Norm_Cumul(Integer(Norm_Cumul(I) * 1000.0)) := Float(I)/100.0;  
        end loop;  
  
        Inv_Norm_Cumul(0) := 0.0;  
        Inv_Norm_Cumul(1000) := 1.0;
```

```

for I in 1..1000 loop
  if Inv_Norm_Cumul(I)=0.0 then
    Inv_Norm_Cumul(I) := Inv_Norm_Cumul(I-1);
  end if;
end loop;

-- for I in 0..100 loop
--   Put_Line(Integer'Image(I*10) & " " &
Float'Image(Inv_Norm_Cumul(I*10)));
-- end loop;
end Calculate_Norm_Cumul;

Gn: Generator;

function Generate_Normal(Mu, Sigma: Float) return Float is

  -- outside 5 sigma on each side of the mean (mu)
  -- the probability is assigned to 0 and then the distribution
  -- is normalized by dividing the density by the total area

begin
  return Mu + Sigma * Inv_Norm_Cumul(Integer(Random(Gn)*1000.0));
end Generate_Normal;

}

```

This macro is invoked in definition 26.

## 7.3 Function and procedure overview

*Function overview*[25] ≡

```

{

  procedure Calculate_All_Trajectories(D: in out Case_Database;
    N_Tasks: Integer:= 1);

  -- -- output parameters

  function Average_Inr(Icase: Integer:= 1;
    From: Integer:= Durmax-24;
    To: Integer:= Durmax-1) return Float;

  function Simple_Raise_Time(Icase: Integer:= 1;
    Raise_Ratio: Float:= 0.95;
    From: Integer:= 0;
    To: Integer:= Durmax-1) return Float;

  function S_Warfarin_Clearance(Icase: Integer:= 1; From: Integer:= 48) return
Float;

  procedure Average_Inr_By_Wes_Wv;
  procedure Raise_Time_By_Wes_Wv(Warf_Dose: Float:= 5.0);
  procedure Bayes_Correction_Experiment_MC;

```

```

function Generate_Normal(Mu, Sigma: Float) return Float;

type Fvector is array(Integer range <>) of Float;
type Ivector is array(Integer range <>) of Integer;

type Case_Coordinates is array(Integer range <>, Integer range <>) of Float;
type Case_Value is array(Integer range <>, Integer range <>) of Float;

-- -- population generators

procedure Generate_Uniform_Wes_Wv(Ces_Min:      Float:= WCESMI;
                                   Ces_Max:      Float:= WCESMX;
                                   Cer_Min:      Float:= WCERMI;
                                   Cer_Max:      Float:= WCERMX;
                                   Cv_Min:       Float:= GCVMI;
                                   Cv_Max:       Float:= GCVMX;
                                   Wes_N,Wv_N:   Integer:= 10);

procedure Generate_Constant_Daily_Dose(Dose: Float:= 1.0; From_Day: Float:= 0.0);

procedure Generate_Random_Cases(Nc: Integer:= 100; Male_Proportion: Float:= 0.5);

-- procedures used for calibration

procedure Find_Doses_For_Target(Target_Inr: Float:= 2.5;
                                Tolerance: Float:= 0.25);

procedure Dangelo_2002_Experiment(N_Cases: Integer:= 9);

-- i/o functions

procedure Save_Db(Filename: String:= "pwissdb.dat");
procedure Save_Signals(Inr, F7, F10, F2, Fr, Fs, Fv, Fh: Boolean:= False;
                      Inr_Name: String:= "pwissinr.dat";
                      F7_Name: String:= "pwissf7.dat";
                      F10_Name: String:= "pwissf10.dat";
                      F2_Name: String:= "pwissf2.dat";
                      Fr_Name: String:= "pwissfr.dat";
                      Fs_Name: String:= "pwissfs.dat";
                      Fv_Name: String:= "pwissfv.dat";
                      Fh_Name: String:= "pwissfh.dat"
                      );
}

```

This macro is invoked in definition 26.

## 7.4 Compilation units overview

This section generates the `pwissimod.ada` file from which an executable is produced by compilation (see below, at compiling and running).

`pwissimod.ada`[26]  $\equiv$

```

{
package Pwiss_Parameters is

    Genetic variability distribution parameters[4]

    Anthropometric variability distribution parameters[10]

end Pwiss_Parameters;

with Pwiss_Parameters; use Pwiss_Parameters;
package Pwiss_General is

    Simulated case database[11]

    Bayesian experiment variables[19]

    Output switch variables[21]

    System variables[23]

    Function overview[25]

    Internal_Inconsistency: exception;

end Pwiss_General;

with Ada.Calendar; use Ada.Calendar;
with ada.numerics.elementary_functions;
use ada.numerics.elementary_functions;
with Text_Io; use Text_Io;
with Ada.Numerics; use Ada.Numerics;
with Ada.Numerics.Float_Random;
use Ada.Numerics.Float_Random;
with Ada.Command_Line; use Ada.Command_Line;

package body Pwiss_General is

    INR trace simulator[13]

    Functionals on the case space[14]

    Utility functions[24]

    Generate constant daily dose[1]

    Monte Carlo generator of case parameters[12]

    Calibration experiments[15]

    Bayesian experiment procedure[20]

    General output procedures[22]

```

```

begin
  Calculate_Norm_Cumul;
end Pwiss_General;

with Ada.Text_Io; use Ada.Text_Io;
with Pwiss_General; use Pwiss_General;
with Ada.Exceptions; use Ada.Exceptions;

procedure Pwisstest is

begin
  Put_Line("This may take over one hour on a 2GHz processor, please be patient...");
  -- Put_Line("Sconce:2005 experiment (297 cases)");
  Generate_Random_Cases(297);
  Find_Doses_For_Target;
  Save_Db;

  -- Put_Line("D'Angelo:2002 experiment (9 cases)");
  Dangelo_2002_Experiment(9);

  -- Put_Line("Bayes correction experiment");
  Bayes_Correction_Experiment_Mc;

exception
  when Error: others =>
    Put_Line( "Erroneous execution:" & Ascii.Lf &
              Exception_Name(Error) & Ascii.Lf & Exception_Message(Error) &
Ascii.Lf
              & Exception_Information(Error) & "</PRE>");
end Pwisstest;

}

```

This macro is attached to an output file.

## 8 Graphics and calibration report

This section produces the popwiss.R file that you need to run through a recent R statistical system package in order to produce graphics from simulation output. All graphics and tables in this document are produced by calling functions from the program below, as seen at the end of the module and exemplified in the **Makefile**.

popwiss.R[27] ≡  
{

```

dens2tile <- function(dens,ntile=100) {
  ddens <- 1/ntile
  cumu <- sapply(1:length(dens), function(x) sum(dens[1:x]))
  tile <<- rep(NA,ntile)
  j <<- 1

```

```

for (i in 1:ntile) {
  while(j<=length(cumu) && cumu[j] < (i* ddens)) j <- j+1
  if (j>=length(cumu))
    tile[i] <- 100
  else {
    if (j==1)
      tile[i] <- 0
    else
      tile[i] <- (j-1) + (i* ddens - cumu[j-1]) / (cumu[j] - cumu[j-1])
  }
}
tile
}

densmat <- function(den) {
  dm <- matrix(sapply(1:80, function(y) dens2tile(den[y*6,])/10),nrow=100)
  dm
}

inrtraceplot <-
function(iftempl="data/bayexpmc_inr_limits.dat",ifnum=1,withsim=TRUE,col="red") {
  ifn <- paste(iftempl, ifnum, sep="")
  ifdat <- read.table("data/bayexpmc_inr_series.dat",header=TRUE)
  dm1 <- densmat(as.matrix(read.table(ifn)))
  plot( as.vector(dm1[1,1:80]), ylim=c(0,10),
        type="l", col="black", lty="24", xlab="6 hour intervals", ylab="INR")
  lines(dm1[99,],type="l",col="black",lty="24")
  lines(dm1[15,],type="l",col="black")
  lines(dm1[85,],type="l",col="black")
  lines(dm1[50,],type="l",col="black",lwd=2)

  points(ifdat$T[ifdat$MSET==ifnum]/6,ifdat$INR[ifdat$MSET==ifnum],pch="+",col="black")
  # lines(ifdat$T[ifdat$MSET==ifnum]/6,ifdat$INRO[ifdat$MSET==ifnum],col="red")
  if(withsim) {
    inr <- scan(paste("data/inr_",ifdat$SIM[ifdat$MSET==ifnum][1],sep=""))
    lines(1:480/6,inr[1:480],col=col,lty="88") # was "red"
  }
}

pnas.f2bs <- function(maxi) {
  for (fi in 1:maxi) {
    postscript(paste("data2/inrp",fi,".eps",sep=""))
    par(cex=2)
    inrtraceplot(ifnum=fi)
    dev.off()
  }
}

pnas.f2a <- function() {
  postscript("data2/inrp0.eps")
  par(cex=2)
  inrtraceplot("data/apriori_inr_limits.dat","",FALSE)
  dev.off()
}

```

```

}

pnas.f2 <- function(ifn="data/bayexpmc_inr_limits.dat") {

  # 8.7 cm = 3.425in; x 1200dpi = 4110
  # golden ratio: 1.618 x 4110 = 6650

  nu <- as.matrix(read.table(ifn))
  postscript("data2/f2pnas.ps") #,width=3.425,height=3.425*1.618)
  par(mfrow=c(3,2),cex=1)
  for(i in 2:7) {
    plot(1:100/10,nu[60*i,],xlim=c(1,4),
         ylim=c(0,0.4),type="l",ylab="p.density",xlab="INR")
  }
  dev.off()
}

pnas.f2cs <- function(ifn="data/bayexpmc_postprob.dat",nfiles=50) {
  for(fni in 1:nfiles) {
    pw <- matrix(scan(paste(ifn,fni,sep="")),nrow=2500)
    postscript(paste("data2/fig2c",fni,".eps",sep=""))
    #,width=3.425,height=3.425*1.618)
    par(mfrow=c(3,3),cex=1,mar=c(1,1,1,1))
    for(i in 1:9) {
      contour(matrix(pw[,i*2-1],ncol=50), # window used only in the main text
              figure: [10:40,30:40],
              drawlabels=FALSE,
              axes=FALSE,frame.plot=TRUE,levels=0:20*0.002,zlim=c(min(pw),max(pw)),
              xlab="CYP2C9",ylab="VKLORC1")
    }
    dev.off()
    # postscript(paste("data2/cf3pnas",fni,".eps",sep=""),horizontal=FALSE)
    #,width=3.425,height=3.425*1.618)
    # par(mfrow=c(5,2),cex=1,mar=c(2,2,2,2))
    # for(i in 1:10) {
    #   filled.contour(matrix(pw[,i],ncol=50),zlim=c(min(pw),max(pw)))
    # }
    # dev.off()
  }
}

pnas.f2 <- function(nfiles=50) {
  pnas.f2a()
  pnas.f2bs(nfiles)
  pnas.f2cs(nfiles=nfiles)
  system(paste("cd data2 ; epstopdf inrp0.eps",sep=""))
  for(i in 1:nfiles) {
    system(paste("cd data2 ; epstopdf inrp",i,".eps",sep=""))
    system(paste("cd data2 ; epstopdf fig2c",i,".eps",sep=""))
    sink(paste("data2/fig2-",i,".tex",sep=""))
    cat(paste("\\documentclass{article}
\\usepackage{graphicx}
\\usepackage[scale=0.99]{geometry}

```

```

\\begin{document}\\pagestyle{empty}
\\includegraphics[height=8.7cm,angle=-90]{inrp0}\\par
\\includegraphics[height=8.7cm,angle=-90]{inrp",i,"}\\par
\\includegraphics[height=8.7cm,angle=-90]{fig2c",i,"}
\\end{document}
",sep="")
  sink()
  system(paste("cd data2 ; pdflatex -halt-on-error fig2-",i,".tex",sep=""))
}
}

```

*estimate BMI dispersion*[9]

# procedures to reproduce results from [2]

# figure 1

```

f1.sconce.blood2005 <- function () {
  di <- read.table("data/pwissdb.dat", header=TRUE)
  sset <- di$n
  boxplot(list(AA11=di$avgdose[sset][di$CYP2C9[sset]=="CYP2C9_1_1" &
di$VKORC1[sset]=="VKORC1_A_A"],
  GA11=di$avgdose[sset][di$CYP2C9[sset]=="CYP2C9_1_1" &
di$VKORC1[sset]=="VKORC1_G_A"],
  GG11=di$avgdose[sset][di$CYP2C9[sset]=="CYP2C9_1_1" &
di$VKORC1[sset]=="VKORC1_G_G"],
  AA12=di$avgdose[sset][di$CYP2C9[sset]=="CYP2C9_1_2" & di$VKORC1[sset]=="VKORC1_A_A"],
  GA12=di$avgdose[sset][di$CYP2C9[sset]=="CYP2C9_1_2" &
di$VKORC1[sset]=="VKORC1_G_A"],
  GG12=di$avgdose[sset][di$CYP2C9[sset]=="CYP2C9_1_2" &
di$VKORC1[sset]=="VKORC1_G_G"],
  AA13=di$avgdose[sset][di$CYP2C9[sset]=="CYP2C9_1_3" & di$VKORC1[sset]=="VKORC1_A_A"],
  GA13=di$avgdose[sset][di$CYP2C9[sset]=="CYP2C9_1_3" &
di$VKORC1[sset]=="VKORC1_G_A"],
  GG13=di$avgdose[sset][di$CYP2C9[sset]=="CYP2C9_1_3" &
di$VKORC1[sset]=="VKORC1_G_G"],
  AA22=di$avgdose[sset][di$CYP2C9[sset]=="CYP2C9_2_2" & di$VKORC1[sset]=="VKORC1_A_A"],
  GA22=di$avgdose[sset][di$CYP2C9[sset]=="CYP2C9_2_2" &
di$VKORC1[sset]=="VKORC1_G_A"],
  GG22=di$avgdose[sset][di$CYP2C9[sset]=="CYP2C9_2_2" &
di$VKORC1[sset]=="VKORC1_G_G"],
  AA23=di$avgdose[sset][di$CYP2C9[sset]=="CYP2C9_2_3" & di$VKORC1[sset]=="VKORC1_A_A"],
  GA23=di$avgdose[sset][di$CYP2C9[sset]=="CYP2C9_2_3" &
di$VKORC1[sset]=="VKORC1_G_A"],
  GG23=di$avgdose[sset][di$CYP2C9[sset]=="CYP2C9_2_3" &
di$VKORC1[sset]=="VKORC1_G_G"],
  AA33=di$avgdose[sset][di$CYP2C9[sset]=="CYP2C9_3_3" & di$VKORC1[sset]=="VKORC1_A_A"],
  GA33=di$avgdose[sset][di$CYP2C9[sset]=="CYP2C9_3_3" &
di$VKORC1[sset]=="VKORC1_G_A"],
  GG33=di$avgdose[sset][di$CYP2C9[sset]=="CYP2C9_3_3" &
di$VKORC1[sset]=="VKORC1_G_G"]),
  ylim=c(0,12))

```

```

}

# figure 2

f2.sconce.blood2005 <- function () {
  di <- read.table("data/pwissdb.dat", header=TRUE)

  estdose <- 0.628 - 0.0135 * 70 - 0.240 * (di$CYP2C9=="CYP2C9_1_2" |
    di$CYP2C9=="CYP2C9_2_2" | di$CYP2C9=="CYP2C9_2_3") -
    0.240 * (di$CYP2C9=="CYP2C9_2_2") - 0.370 * (di$CYP2C9=="CYP2C9_1_3" |
    di$CYP2C9=="CYP2C9_2_3" | di$CYP2C9=="CYP2C9_3_3") -
    0.370 * (di$CYP2C9=="CYP2C9_3_3") - 0.241 * (di$VKORC1=="VKORC1_G_G") -
    - 2 * 0.241 * (di$VKORC1=="VKORC1_G_A") - 3 * 0.241 * (di$VKORC1=="VKORC1_A_A")
+
  1.62 * di$height

  estdose <- estdose * estdose

  plot(estdose[100:138],di$avgdose[100:138],xlab="calculated warfarin
dose",ylab="actual warfarin dose(mg)",
  xlim=c(0,8),ylim=c(0,8))
}

# results as in [2], section 'associations with warfarin dose'

mean.sd.dose <- function(di,sset) {
  paste(formatC(mean(di$avgdose[sset]),width=5), " (",
    formatC(sd(di$avgdose[sset],na.rm=TRUE),width=5), ")", sep="")
}

s.dose.assoc.sconce2005 <- function() {
  di <- read.table("data/pwissdb.dat", header=TRUE)

  cat("Warfarin dose association summary\n")
  cat("compared to Sconce et al, blood 2005, 206:2329\n\n")
  cat("subset          warfarissimo-2          sconce:2005\n")
  cat(" (sex)\n")
  cat("females          ",
formatC(median(di$avgdose[di$sex=="F"],na.rm=TRUE),width=5),"          2.9 \n")
  cat("males          ", for-
matC(median(di$avgdose[di$sex=="M"],na.rm=TRUE),width=5),"          3.7 \n\n")
  cat(" (CYP2C9)\n")
  cat("*1*1          ", mean.sd.dose(di,di$CYP2C9=="CYP2C9_1_1"),"          4.08
(2.13)\n")
  cat("*1*2          ", mean.sd.dose(di,di$CYP2C9=="CYP2C9_1_2"),"          3.56
(1.82)\n")
  cat("*1*3          ", mean.sd.dose(di,di$CYP2C9=="CYP2C9_1_3"),"          2.70
(1.38)\n")
  cat("*2*2          ", mean.sd.dose(di,di$CYP2C9=="CYP2C9_2_2"),"          1.92
(1.12)\n")
  cat("*2/3*3          ", mean.sd.dose(di,di$CYP2C9=="CYP2C9_2_3" |
di$CYP2C9=="CYP2C9_3_3"),
"          1.58 (0.79)\n\n")
}

```

```

        cat(" (VKORC1)\n")
        cat("GG          ", mean.sd.dose(di,di$VKORC1=="VKORC1_G_G"),"          4.53
(2.22)\n")
        cat("GA          ", mean.sd.dose(di,di$VKORC1=="VKORC1_G_A"),"          3.83
(1.91)\n")
        cat("AA          ", mean.sd.dose(di,di$VKORC1=="VKORC1_A_A"),"          2.23
(1.26)\n\n")

    }

# simulation of the results from [6]

dangelo2002.inr.graph <- function() {
  finr <- read.table("factinr.dat",header=TRUE)
  par(mfrow=c(2,2))

  dang.inr <- as.matrix(read.table("data/dangelo2000_inr.dat"))
  plot(dang.inr[1,], type="l", ylim=c(0,5),xlab="hours",ylab="INR")
  for (i in 2:dim(dang.inr)[1])
    lines(dang.inr[i,], type="l")
  lines(sapply(1:dim(dang.inr)[2], function(x) mean(dang.inr[,x])), col="magenta")
  lines(132+24*(0:15),finr$INR,col="red")
  dang.fh <- as.matrix(read.table("data/dangelo2000_fh.dat"))
  lines(sapply(1:dim(dang.fh)[2], function(x) mean(dang.fh[,x])), col="green")
  dang.fv <- as.matrix(read.table("data/dangelo2000_fv.dat"))
  lines(sapply(1:dim(dang.fv)[2], function(x) mean(dang.fv[,x])),
col="green",lty="88")
  dang.fs <- as.matrix(read.table("data/dangelo2000_fs.dat"))
  lines(sapply(1:dim(dang.fs)[2],
    function(x) 3+10*mean(dang.fs[,x])), col="black",lwd=2,)
  segments(0,3,500,3,col="gray")

  segments(6:13 * 24,0,6:13*24,5,col="gray")

  dang.f2 <- as.matrix(read.table("data/dangelo2000_f2.dat"))
  plot(dang.f2[1,], type="l", ylim=c(0,1.5),xlab="hours",ylab="F.II")
  for (i in 2:dim(dang.f2)[1])
    lines(dang.f2[i,], type="l")
  lines(sapply(1:dim(dang.f2)[2], function(x) mean(dang.f2[,x])), col="magenta")
  lines(144+24*(0:15),1.33* finr$F.II/100,col="red") # 1.33 conciliates watala with
d'angelo

  dang.f7 <- as.matrix(read.table("data/dangelo2000_f7.dat"))
  plot(dang.f7[1,], type="l", ylim=c(0,1.5),xlab="hours",ylab="F.VII")
  for (i in 2:dim(dang.f7)[1])
    lines(dang.f7[i,], type="l")
  lines(sapply(1:dim(dang.f7)[2], function(x) mean(dang.f7[,x])), col="magenta")
  lines(144+24*(0:15),1.33 * finr$F.VII/100,col="red")

  dang.f10 <- as.matrix(read.table("data/dangelo2000_f10.dat"))
  plot(dang.f10[1,], type="l", ylim=c(0,1.5),xlab="hours",ylab="F.X")

```

```

for (i in 2:dim(dang.f10)[1])
  lines(dang.f10[i,], type="l")
lines(sapply(1:dim(dang.f10)[2], function(x) mean(dang.f10[,x])), col="magenta")
lines(144+24*(0:15),1.33 * finr$F.X/100,col="red")

}

# report on reproducibility of experimental studies

exper.rep.report <- function() {
  png("data2/f1sconce2005.png",width=1100,height=800)
  f1.sconce.blood2005()
  dev.off()
  png("data2/f2sconce2005.png",width=800,height=800)
  f2.sconce.blood2005()
  dev.off()
  png("data2/dangelo2002.png",width=800,height=800)
  dangelo2002.inr.graph()
  dev.off()
  sink("data2/expereport.html")
  cat("<html><head><title>Reproduction of experimental data\
  by warfarissimo-2</title></head><body>\n")
  cat("<b>Reproduction of experimental data by the population\
  simulator warfarissimo2 </b>")
  cat("<P>[1] Sconce EA, Khan TI, Wynne HA, Avery P, Monkhouse L, King BP, Wood P,\
  Kesteven P, Daly AK, Kamali F.<BR>\
  The impact of CYP2C9 and VKORC1 genetic polymorphism and\
  patient characteristics upon warfarin dose requirements: proposal for\
  a new dosing regimen.<BR>\
  Blood. 2005 Oct 1;106(7):2329-33.\n")
  cat("<P>Figure 1. Dose needed for a target INR of 2.5 by genotype\
  in 297 subjects. On y, dose in mg/day. On x genotype of\
  both VKORC1 (AA,GA,GG) and CYP2C9 (11,12,13,22,23,33).")
  cat("<img src=\"f1sconce2005.png\"><br>\n")
  cat("<P>Data reported in section 'associations of warfarin dose'\<BR>\n")
  cat("<pre>\n")
  s.dose.assoc.sconce2005()
  cat("</pre>\n\n")
  cat("<P>[2] D'Angelo A, Della Valle P, Crippa L, Fattorini A, Pattarini E, Vigan,
  D'Angelo S\
  <BR>Relationship Between International Normalized Ratio Values,\
  Vitamin K-Dependent Clotting Factor Levels and in Vivo Prothrombin\
  Activation During The Early and Steady Phases of Oral Anticoagulant
  Treatment.<BR>\
  Haematologica. 2002 Oct;87(10):1074-80.")
  cat("<P>The figure below should be comparable with figure 1. Black lines:\
  factors and INRs in each simulated case in time. Magenta line:\
  instantaneous average. Red line: instantaneous average in cases generated as in
  [2]\
  with a female:male ratio of 3:6. Continuous green line: reduced vitamin K.\
  dashed green: active VKORC1; Thick black line: s-warfarin concentration;
  vertical\
  gray bars: instant of warfarin administration.<P>")
}

```

```

cat("<img src=\"dangelo2002.png\"><br>\n")
cat("<P>[3] Shikata E, Ieiri I, Ishiguro S, Aono H, Inoue K, Koide T, Ohgi S,
Otsubo K.<BR>\
  Association of Pharmacokinetic (CYP2C9) and Pharmacodynamic\
  (Factors II, VII, IX, and X; Proteins S and C; and\
  Gamma-Glutamyl Carboxylase) Gene Variants with Warfarin\
  Sensitivity.\
  Blood. 2004 Apr 1;103(7):2630-5. Epub 2003 Dec 4.<BR>\n\n\
  <P><i>Average clearance of total blood S-warfarin is 0.1 ml/min/kg\
  in healthy japanese</i>\
  <P>[4] Takahashi H, Ieiri I, Wilkinson GR, Mayo G, Kashima T, Kimura S,\
  Otsubo K, Echizen H.<BR>\
  5'-Flanking Region Polymorphisms of CYP2C9 and Their\
  Relationship To S-Warfarin Metabolism in White and Japanese\
  Patients.<BR>\
  Blood. 2004 Apr 15;103(8):3055-7. Epub 2003 Dec 30.\
  <P><i>Average clearance of unbound fraction (that is about 0.85% of\
  total s-warfarin) is 10 ml/min/kg in japanese and 5ml/min/kg\
  in healthy european.</i><BR>\
  We conclude average clearance of s-warfarin should be around\
  0.05 ml/min/kg in europeans.<P>")
di <- read.table("data/dangelo2000_db.dat",header=TRUE)
cat("In the simulation of [3], above, is was: ", mean(di$sclr), "(", sd(di$sclr),
")")
cat("</body></html>\n")

}

pnas.inrst <- function() {
  ise <- read.table("data/bayesxpmc_inr_series.dat",header=TRUE)
  png("data2/inrst1.png",width=800,height=800)
  plot(ise$T, ise$INRST, xlab="time (hours)", ylab="INR stability coef")
  dev.off()
  png("data2/inrst2.png",width=800,height=800)
  plot(ise$T[ise$MSET==1], ise$INRST[ise$MSET==1], xlab="time (hours)", ylab="INR
stability coef",
      ylim=c(0,max(ise$INRST)),type="l")
  for (i in 2:50) {
    lines(ise$T[ise$MSET==i], ise$INRST[ise$MSET==i])
  }
  dev.off()
}

sconce.t1 <- function() {
  sink("data2/sconcet1.txt")
  s.dose.assoc.sconce2005()
  sink()
}

sconce.t1()

exper.rep.report()

```

```
pnas.f2()
pnas.inrst()
```

```
}
```

This macro is attached to an output file.

Data from [6]

**factinr.dat**[28] ≡

```
{
  TIME      INR  F.II      F.VII      F.X      P.C
  0          1.00  100      100      100      100
  24         1.15  85       57       67       72
  48         1.55  67       37       55       47
  72         1.90  54       22       38       29
  96         2.15  47       18       27       17
  120        2.30  41       17       21       12
  144        2.62  35       15       17       15
  168        2.55  33       18       15       15
  192        2.50  32       20       14       15
  216        2.20  34       27       16       27
  240        1.62  42       42       22       51
  264        1.15  51       58       41       72
  288        1.02  64       80       57       81
  312        1.00  70       81       64       84
  336        1.00  74       83       68       87
  360        1.00  81       84       75       90
}
```

This macro is attached to an output file.

## 9 Technical issues

### 9.1 System dependencies

In order to produce and run the programs and documentation you need to have a number of packages installed on your system, all of which are freely available on the WWW and are covered by open source licences (mostly the GNU General Public Licence).

- a GNU/Linux distribution such as Ubuntu 7.10
- gnat, the GNU NYU Ada Translator that is the Ada-95 component of the GCC compiler
- R, the statistical package, usually distributed as the r-base and the r-cran debian packages
- funnelweb, a literate programming system
- $\text{\TeX}$ , the standard typesetting system on Linux (including pdf $\text{\TeX}$ )
- make that is usually already installed with linux

## 9.2 Usage

Make a directory such as `~/you/pwiss` containing this file, `pwiss.fw`. `cd` to it and type `fw +T pwiss.fw`. This should produce the following files:

- `pwiss.tex`—source code of the documentation
- `pwissimod.ada`—source code of the simulator
- `popwiss.R`—graphics and validation
- `Makefile`—make file
- `factinr.dat`—data from [6, figure 1].

Then, type `make`. If all packages mentioned at dependencies are installed and everything goes well, after some time (minutes to hours, depending on your processor speed) you will find all the output datafiles in a `data` subdirectory and all graphics in a `data2` subdirectory. Then, `make doc` will produce this document in portable document format.

`Makefile`[29]  $\equiv$

```
{  
  
    all: pwisstest data  
  
    pwissimod.ada: pwiss.fw  
    fw +T pwiss.fw  
  
    pwisstest: pwissimod.ada  
    gnatchop -w -r pwissimod.ada  
    gnatmake pwisstest  
  
    doc: pwiss.fw  
    fw +T pwiss.fw  
    pdftex pwiss  
  
    data: pwisstest popwiss.R  
    mkdir -p data  
    mkdir -p data2  
    cd data; time ../pwisstest  
    cat popwiss.R | R --slave --vanilla  
  
    view:  
    evince pwiss.pdf  
  
}
```

This macro is attached to an output file.

Pwiss was developed by Alexandru Dan Corlan and Marcel Ovidiu Vlad. As all tools and libraries it depends on are covered by the GNU General Public Licence, according to its requirements, this file is also covered by the same licence. While we don't provide the text of the licence here, for concision, it can be easily found on the web and is also included with all Linux distributions (one of which you need to run the program). The individual PK/PD model is based on one previously reported in [8].

## 10 References

1. Ansell, J, Hirsh, J, Poller, L, Bussey, H, Jacobson, A, & Hylek, E. The pharmacology and management of the vitamin K antagonists: the Seventh ACCP Conference on Antithrombotic and Thrombolytic Therapy. (2004) *Chest* **126**, 204S–233S.
2. Sconce, E. A, Khan, T. I, Wynne, H. A, Avery, P, Monkhouse, L, King, B. P, Wood, P, Kesteven, P, Daly, A. K, & Kamali, F. The impact of CYP2C9 and VKORC1 genetic polymorphism and patient characteristics upon warfarin dose requirements: proposal for a new dosing regimen. (2005) *Blood* **106**, 2329–33.
3. Hardmann, J. G & Limbird, L, eds. (2002) *The Pharmacological Basis of Therapeutics*. (McGrawHill).
4. Watala, C, Golanski, J, & Kardas, P. Multivariate relationships between international normalized ratio and vitamin K-dependent coagulation-derived parameters in normal healthy donors and oral anticoagulant therapy patients. (2003) *Thromb J* **1**, 7.
5. Spencer, E. A, Appleby, P. N, Davey, G. K, & Key, T. J. Validity of self-reported height and weight in 4808 EPIC-Oxford participants. (2002) *Public Health Nutr* **5**, 561–5.
6. D’Angelo, A, Valle, P. D, Crippa, L, Fattorini, A, Pattarini, E, & D’Angelo, S. V. Relationship between international normalized ratio values, vitamin K-dependent clotting factor levels and in vivo prothrombin activation during the early and steady phases of oral anticoagulant treatment. (2002) *Haematologica* **87**, 1074–80.
7. Fasco, M. J & Principe, L. M. R- and S-Warfarin inhibition of vitamin K and vitamin K 2,3-epoxide reductase activities in the rat. (1982) *J Biol Chem* **257**, 4894–4901.
8. Corlan, A.D., Corlan, I. & Constantinescu, C. Quantitative consequences of the pharmacological model of oral anticoagulant action, explored through computer simulation (2003) *Terapeutica, Farmacologie si Toxicologie Clinica* **7**, 39–54.

## 11 Results

Graphics generated by the program for 50 runs of the bayesian algorithm, each involving **Case\_Max**=10<sup>5</sup> patients, are shown below. Each run is similar to figures 2b-c from the main text with the difference that, in order to save space, the analogs of figures 2b and 2c are merged together.

The top panel of each run corresponds to figure 2b and depicts the instantaneous distribution of the posterior probability of the INR after each INR determination (each bayesian step). ‘Measured’ (simulated) INR values are represented as crosses. The red line is the ‘actual’ (simulated) INR of one of the cases picked at random from the Monte Carlo generated database, a different one for each run. The thick black line represents the median of the instantaneous INR posterior probability distribution. Continuous thin lines represent the 0.66 confidence intervals. Dotted black lines represent 0.99 confidence intervals.

The bottom panel of each run corresponds to figure 2c from the main paper and represents the marginal probability density distribution of the CYP2C9 and VKORC1 enzyme activities with isodensity lines. The first graph (top left from the group of 9) is identical in all runs and represents the a priori distribution of the activities in the whole simulated population, as resulted from the general calibration. The effect of each bayesian step, after each of the first eight simulated INR measurements, is shown in successive graphs (in lexicographic order). A broader scale is used compared to the example shown in the main paper so that the effect of the algorithm in atypical cases (cases with a combinations of parameters of low probability in the general population) can be observed.

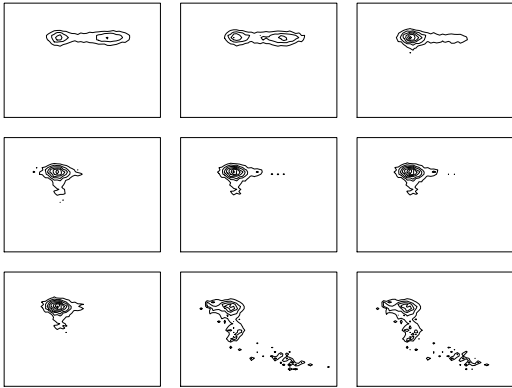
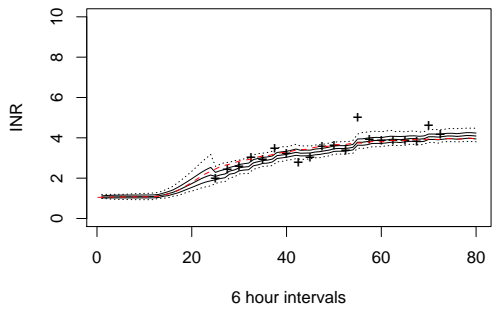


Figure 6.1

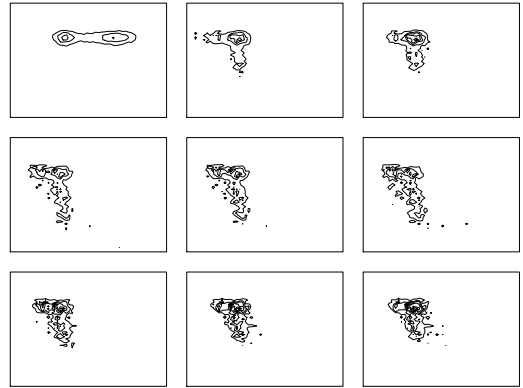
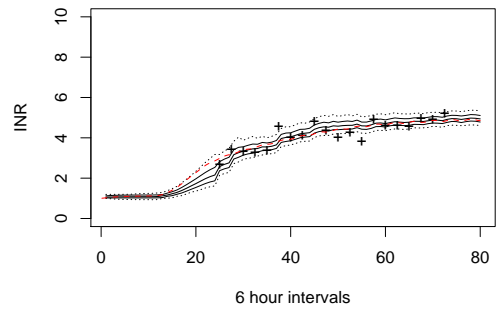


Figure 6.2

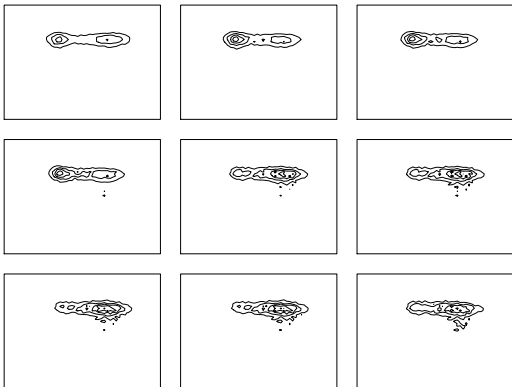
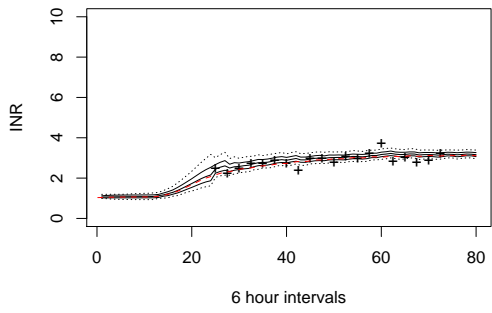


Figure 6.3

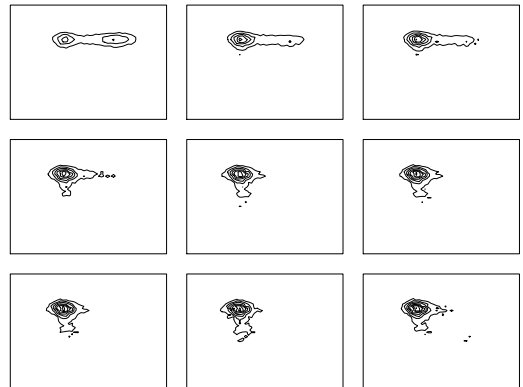
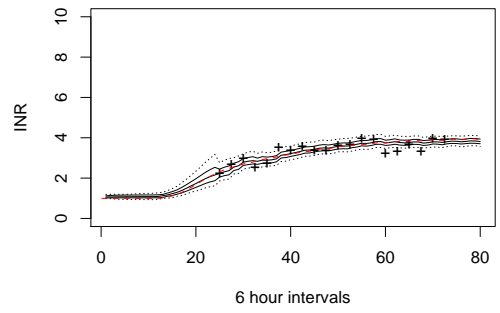


Figure 6.4

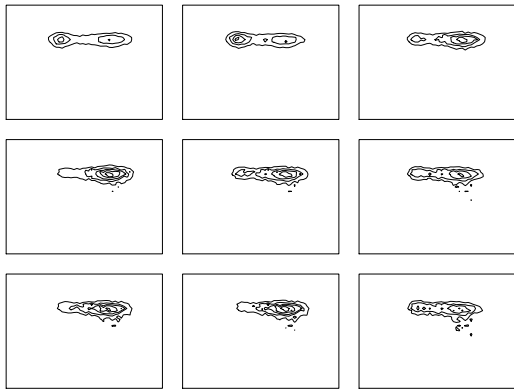
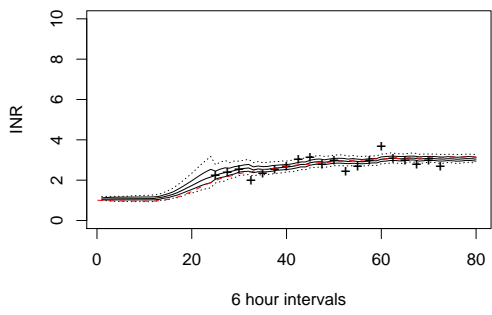


Figure 6.5

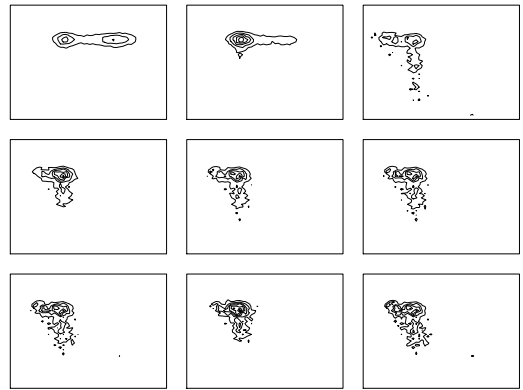
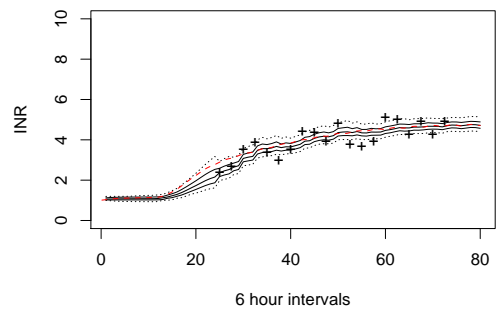


Figure 6.6

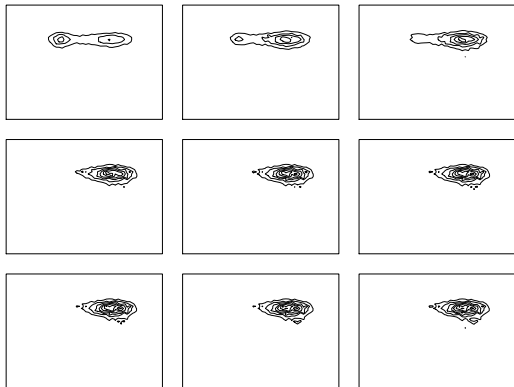
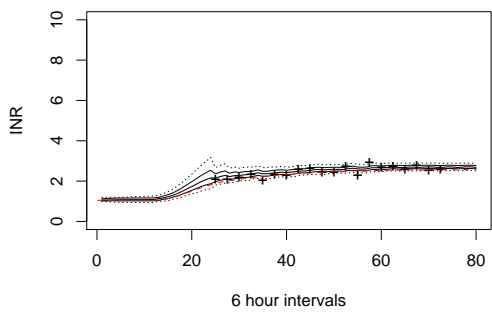


Figure 6.7

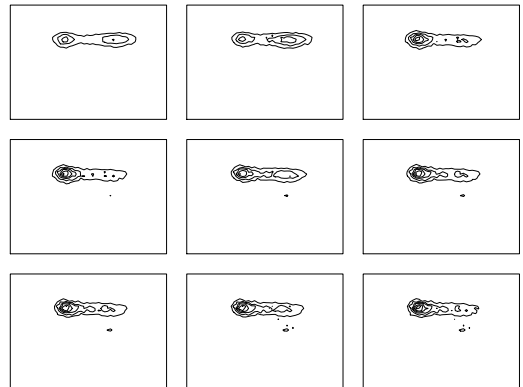
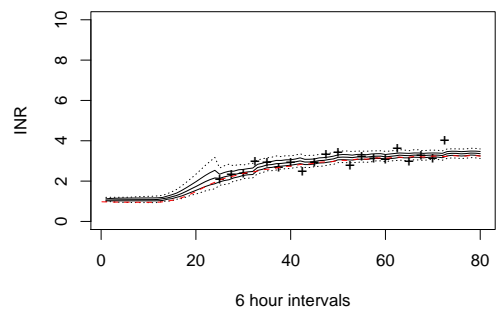


Figure 6.8

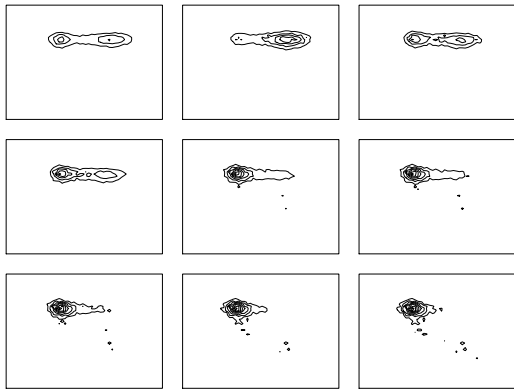
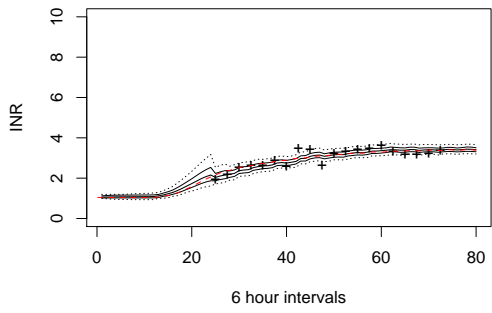


Figure 6.9

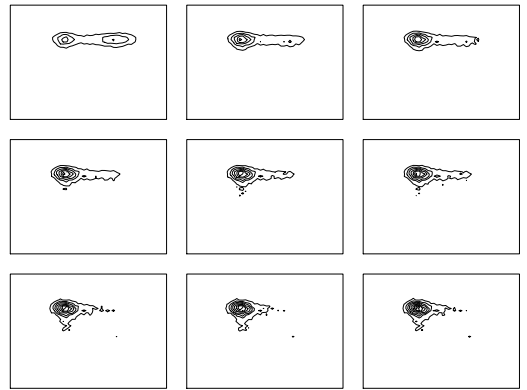
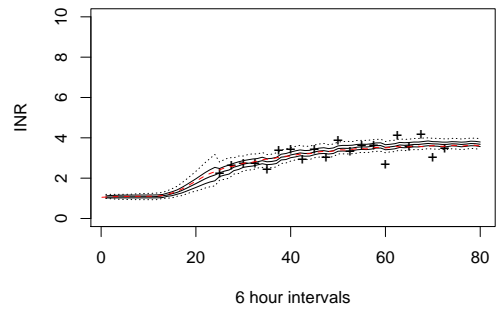


Figure 6.10

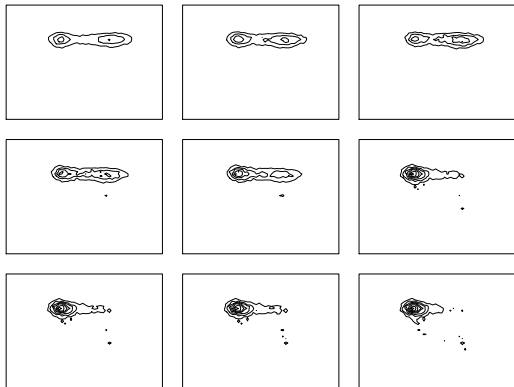
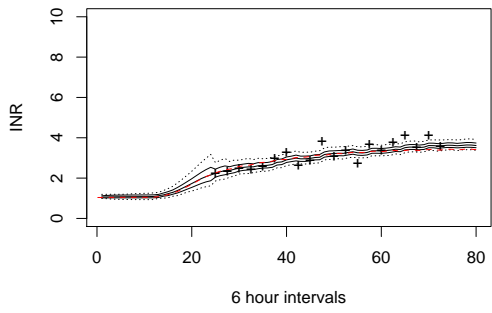


Figure 6.11

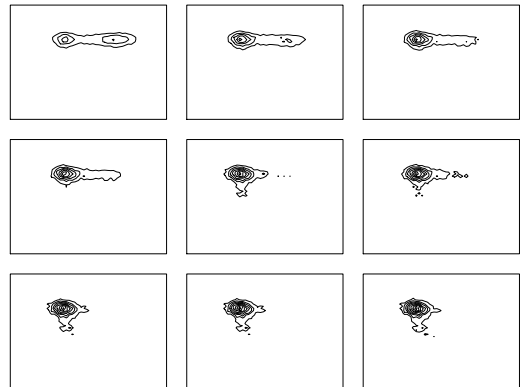
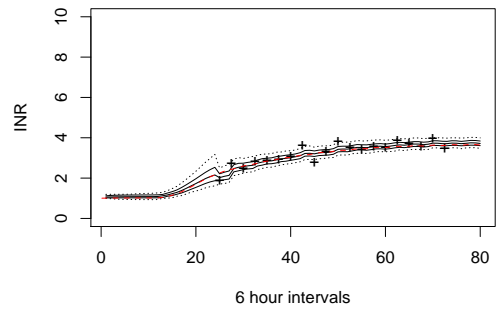


Figure 6.12

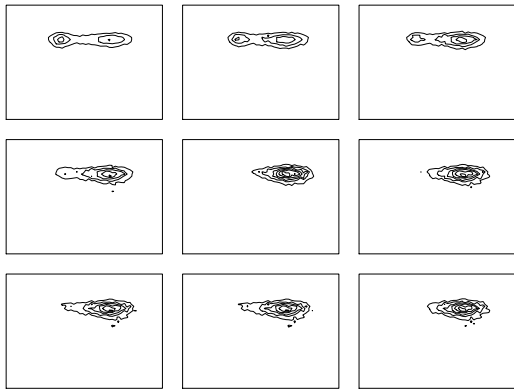
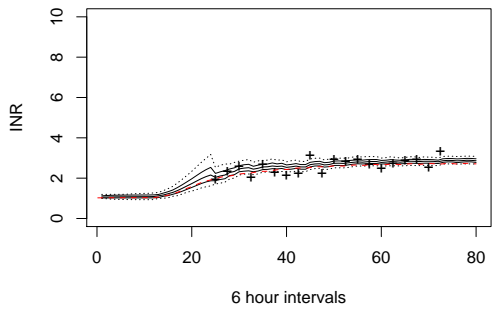


Figure 6.13

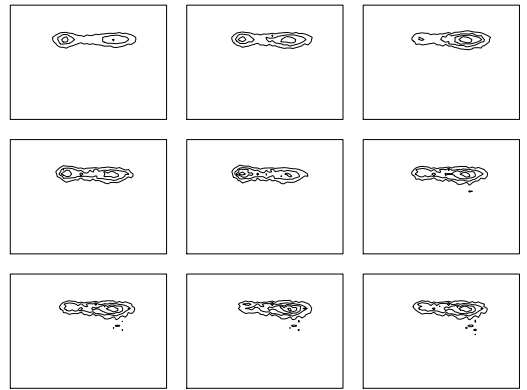
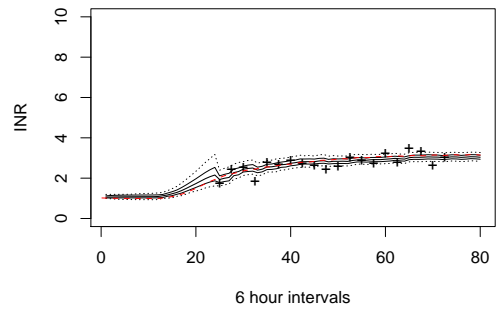


Figure 6.14

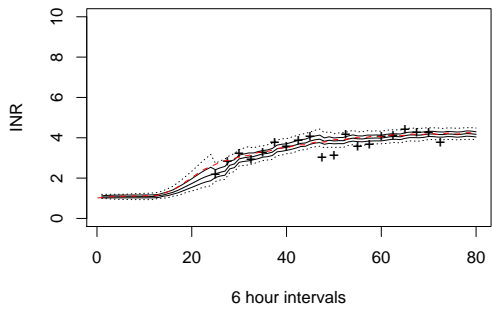


Figure 6.15

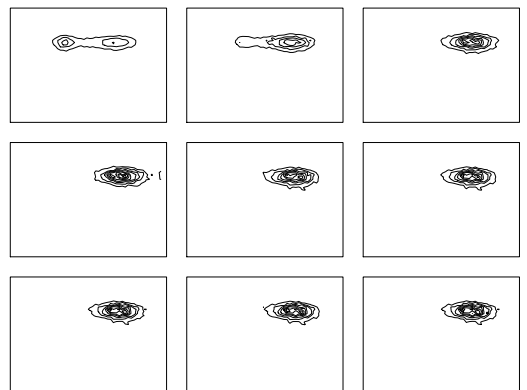
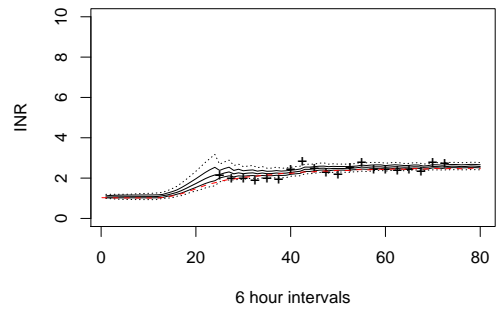


Figure 6.16

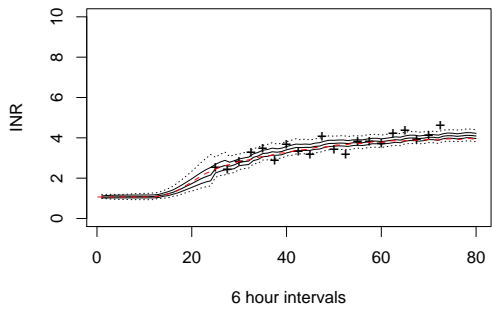


Figure 6.17

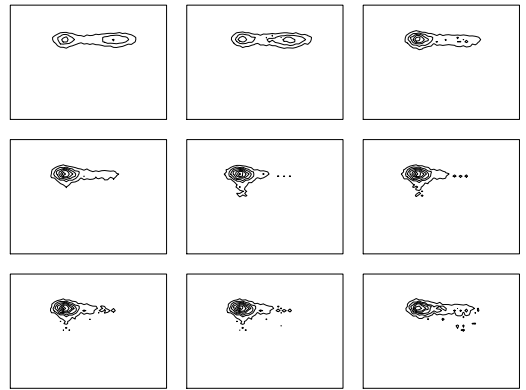
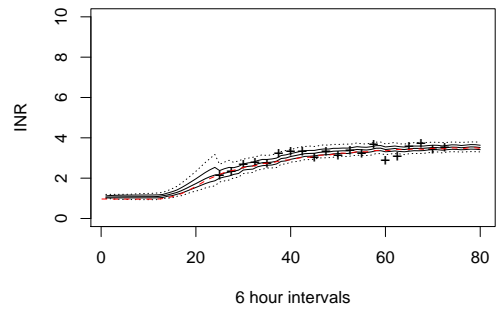


Figure 6.18

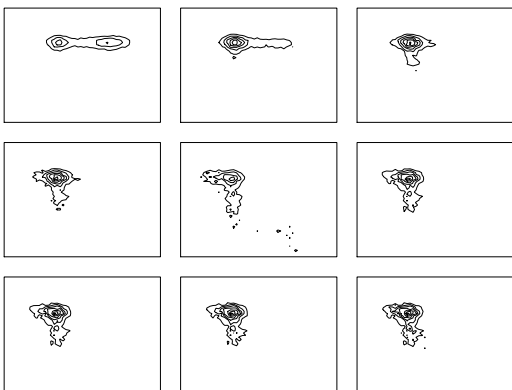
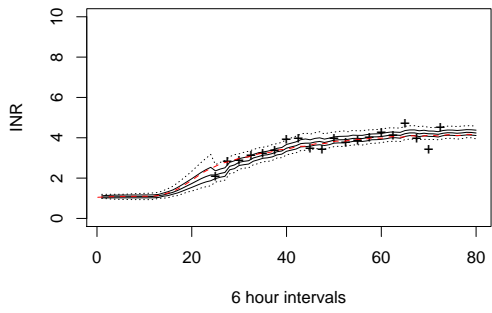


Figure 6.19

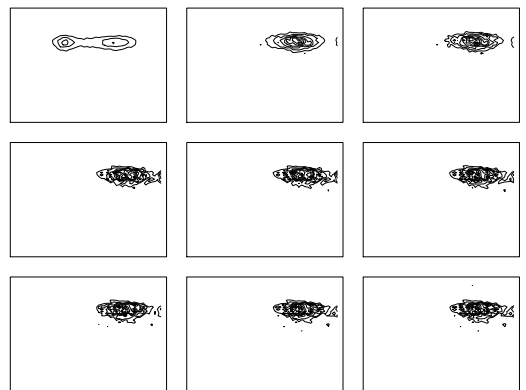
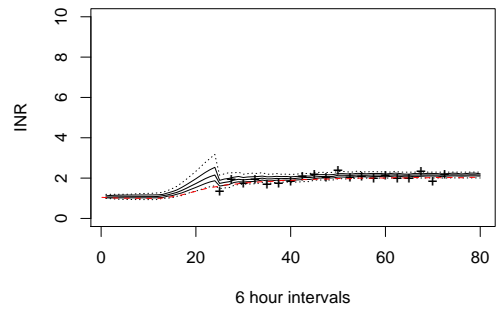


Figure 6.20

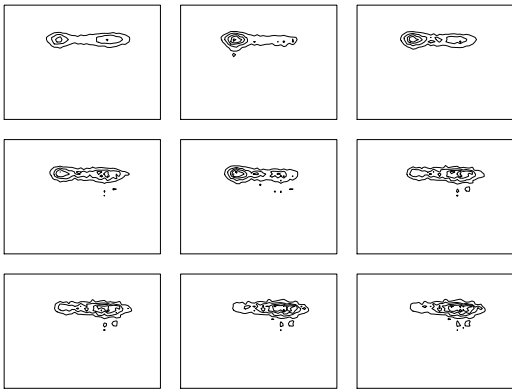
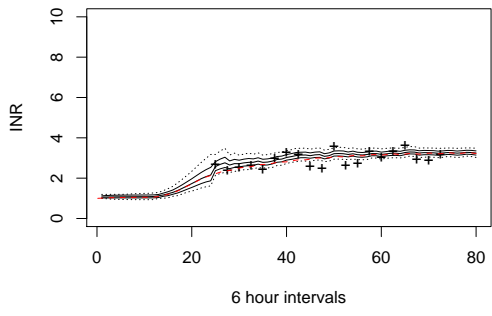


Figure 6.21

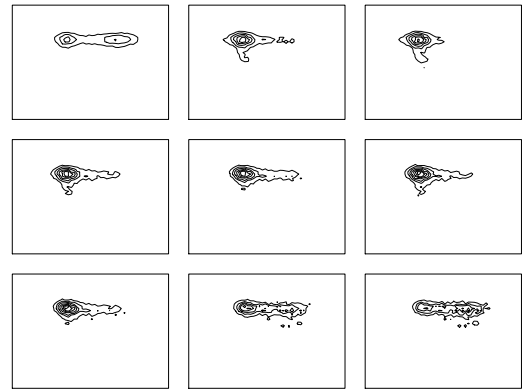
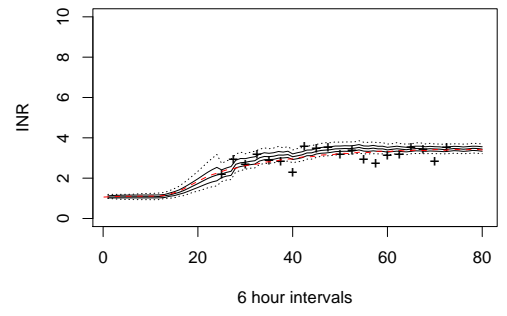


Figure 6.22

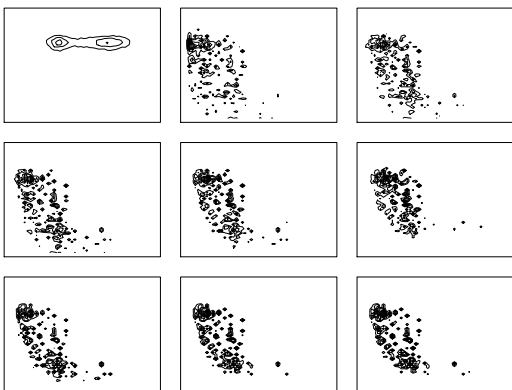
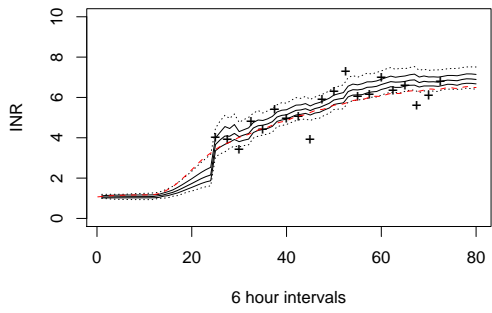


Figure 6.23

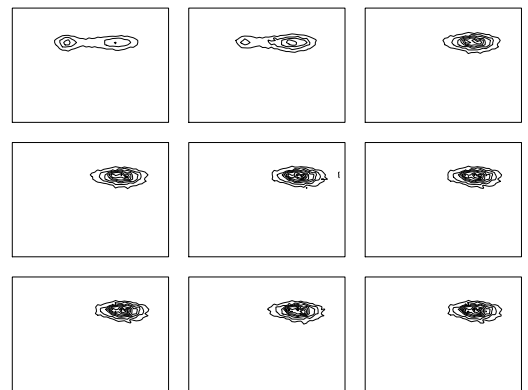
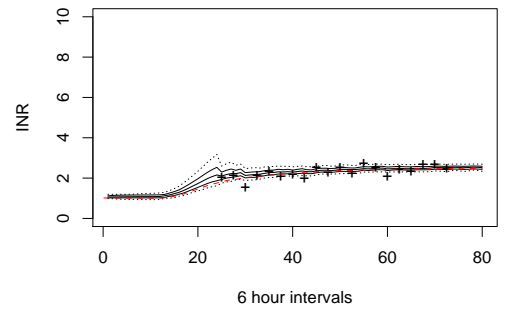


Figure 6.24

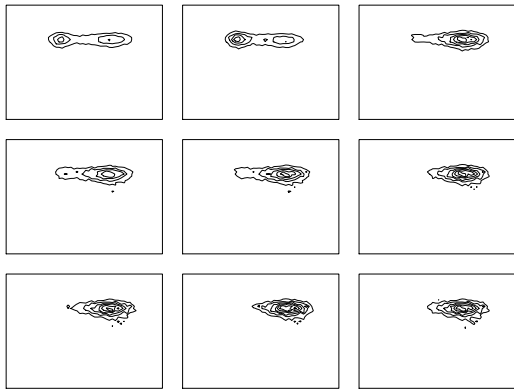
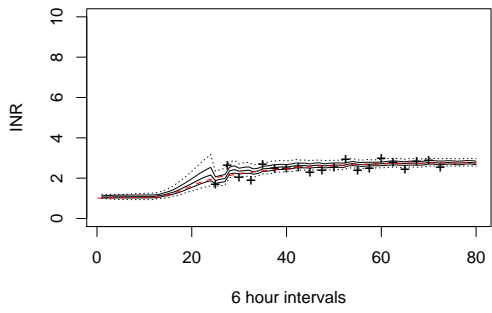


Figure 6.25

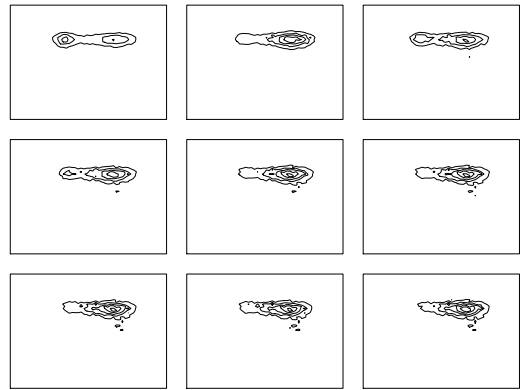
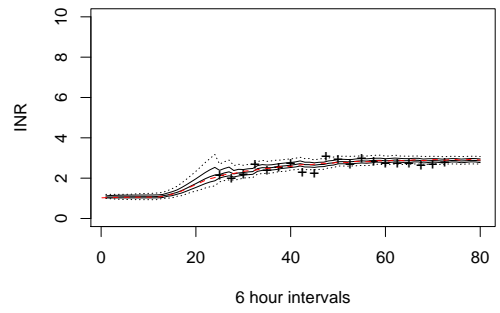


Figure 6.26

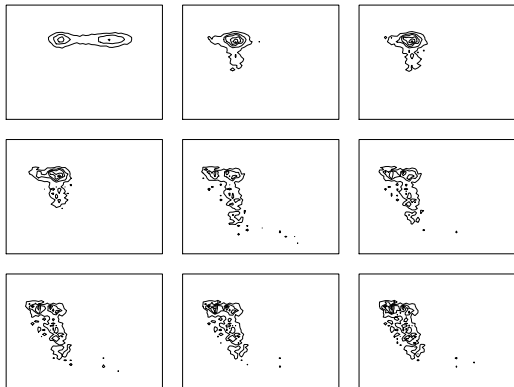
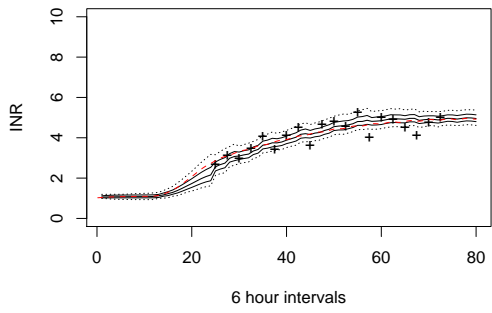


Figure 6.27

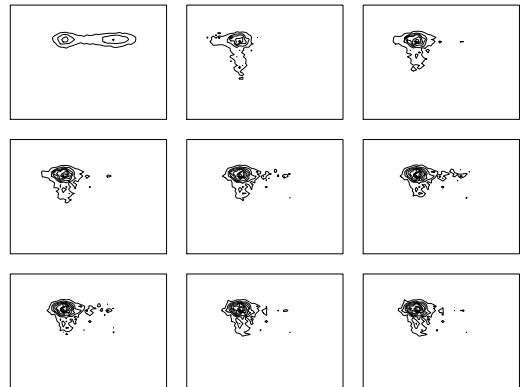
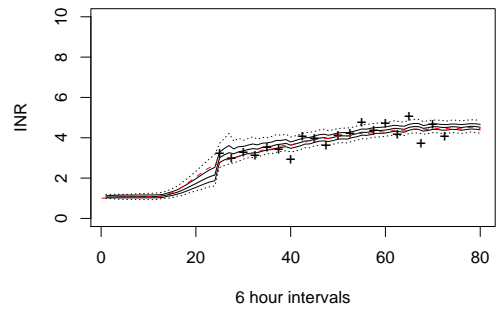


Figure 6.28

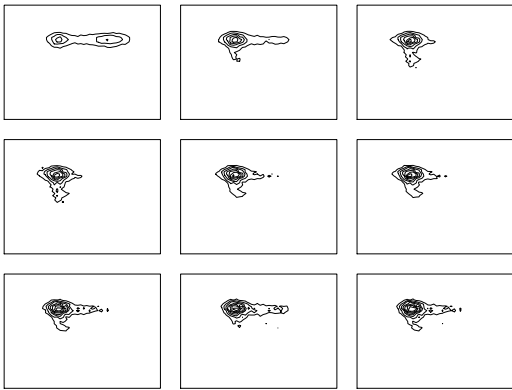
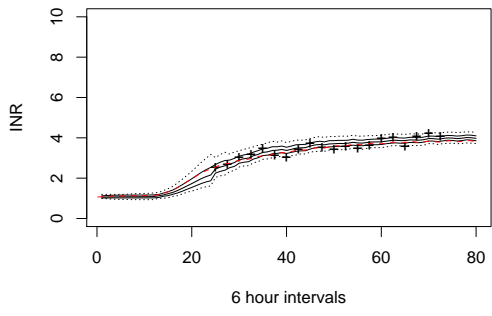


Figure 6.29

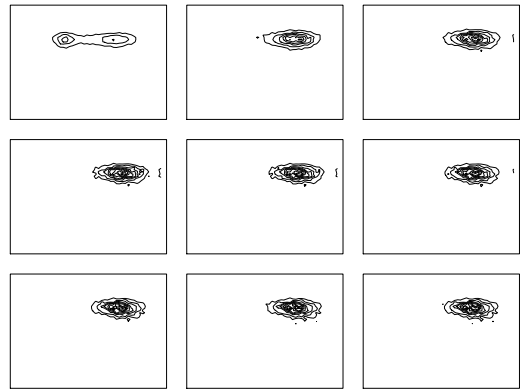
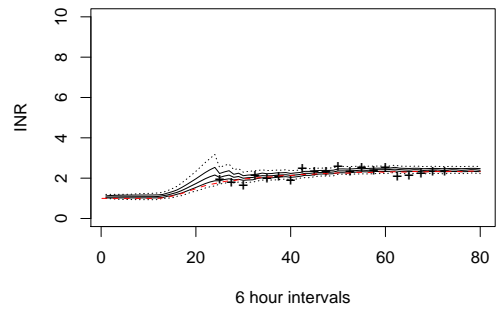


Figure 6.30

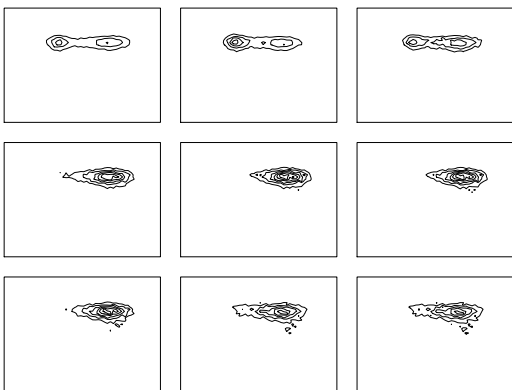
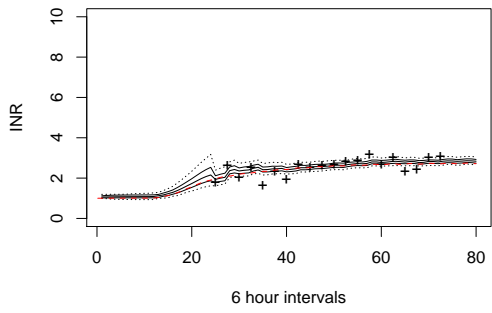


Figure 6.31

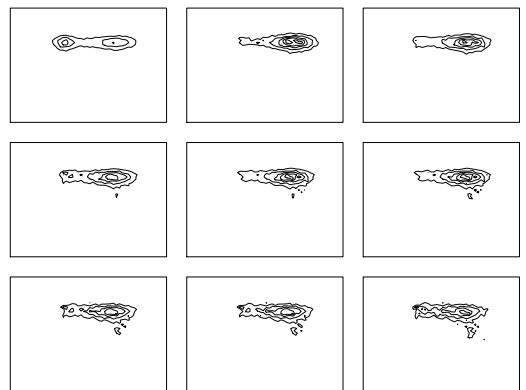
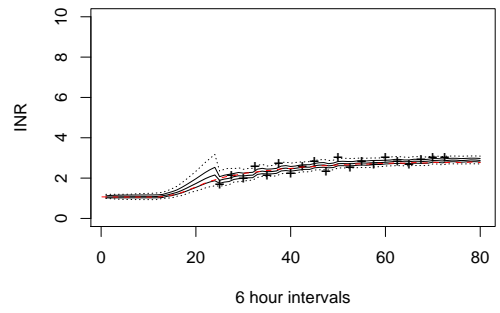


Figure 6.32

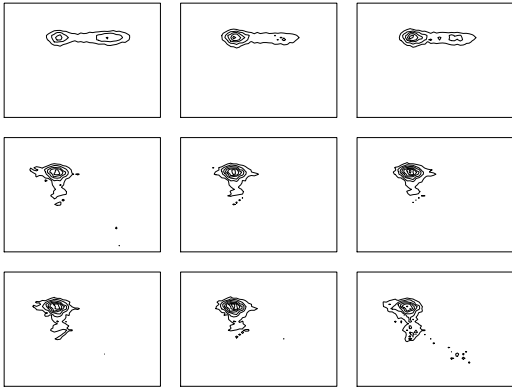
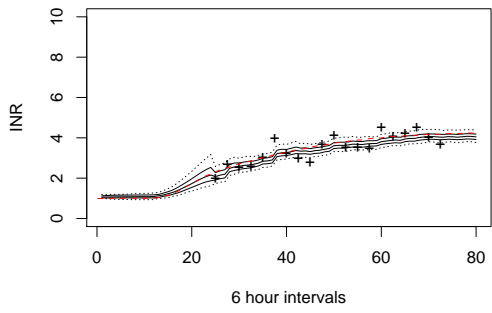


Figure 6.33

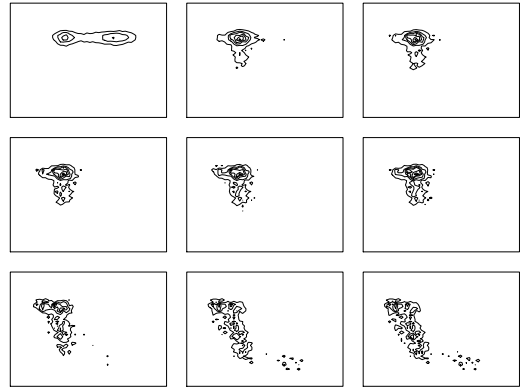
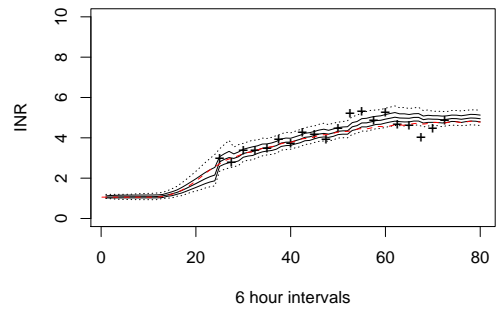


Figure 6.34

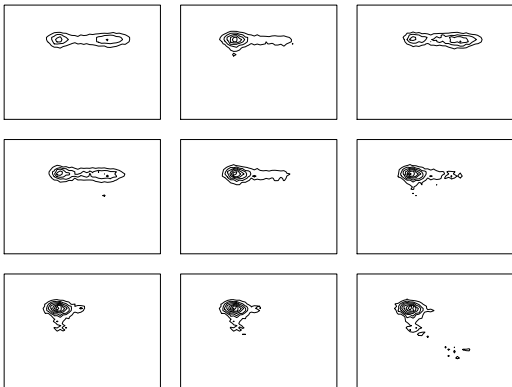
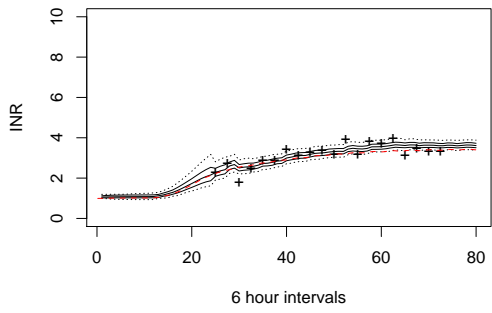


Figure 6.35

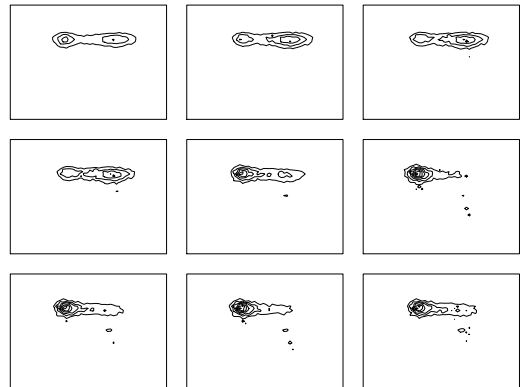
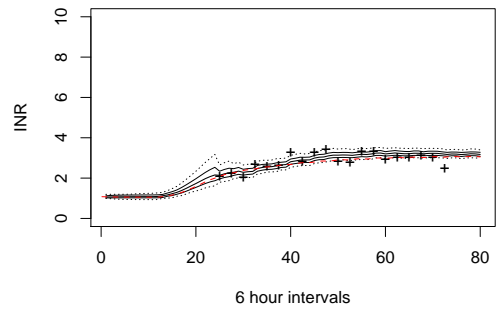


Figure 6.36

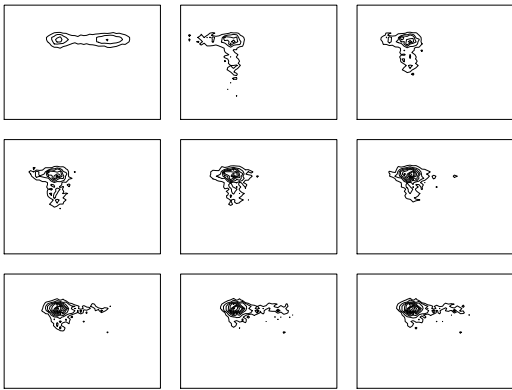
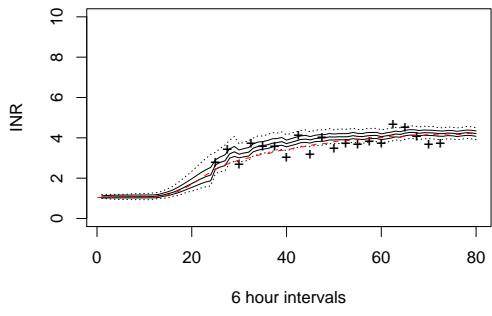


Figure 6.37

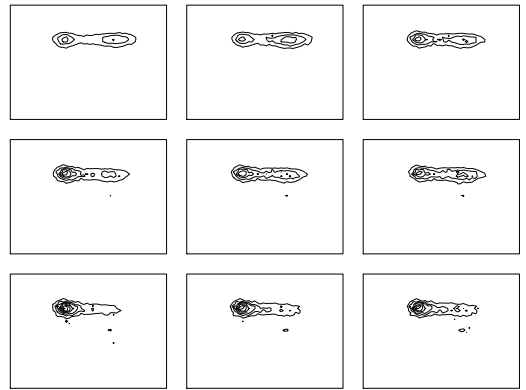
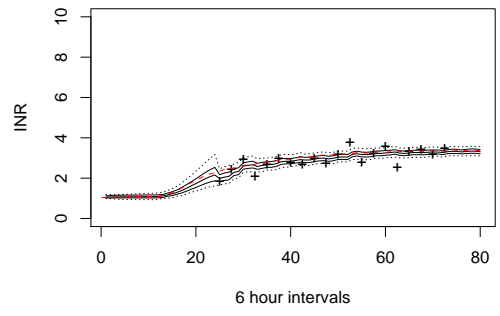


Figure 6.38

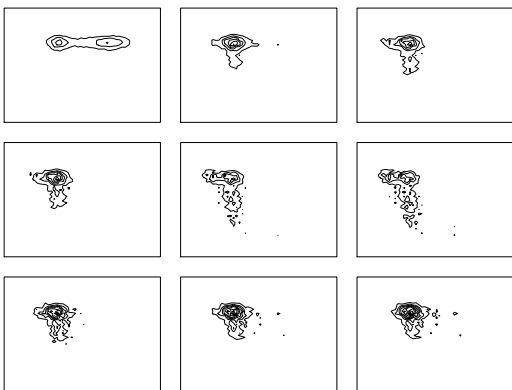
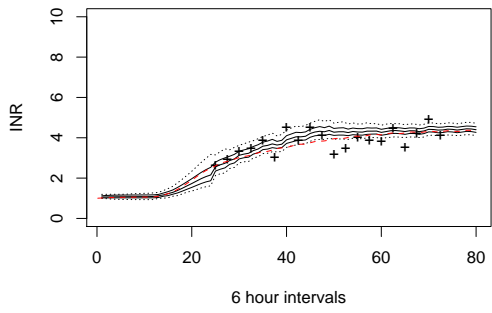


Figure 6.39

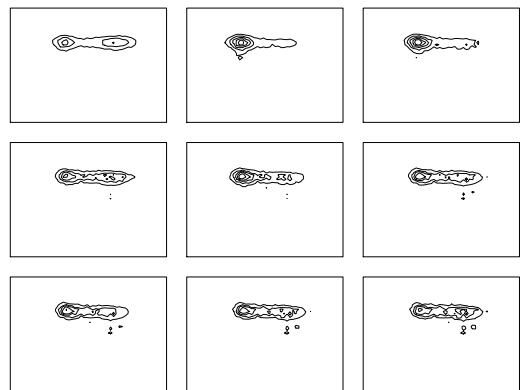
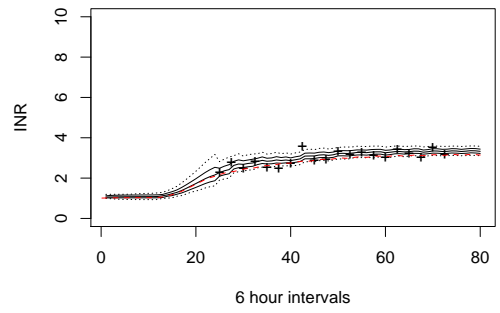


Figure 6.40

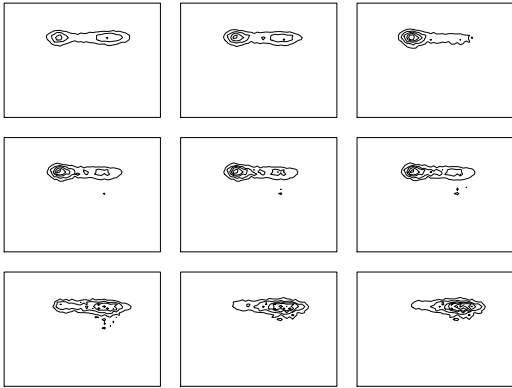
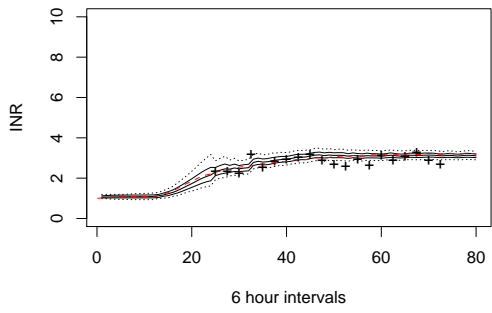


Figure 6.41

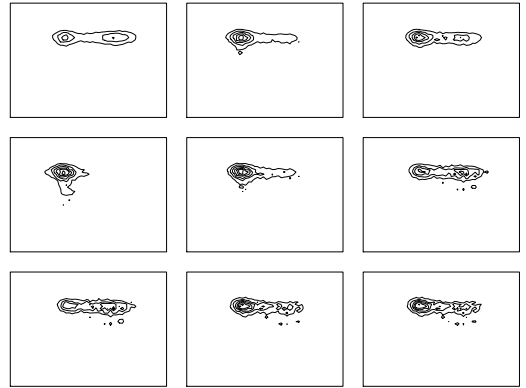
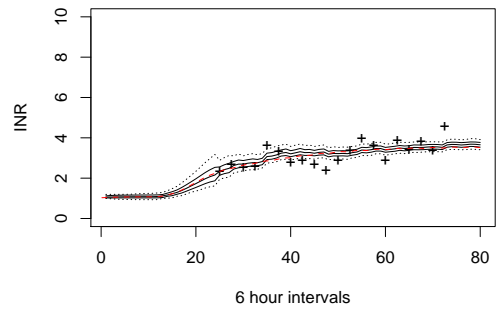


Figure 6.42

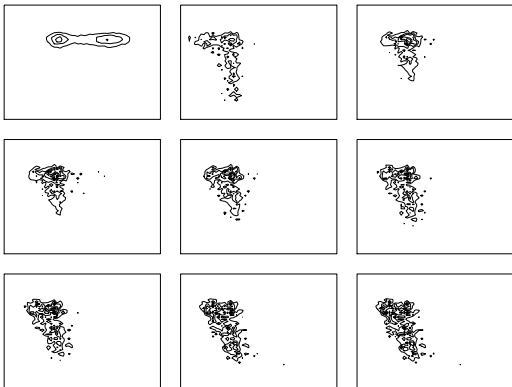
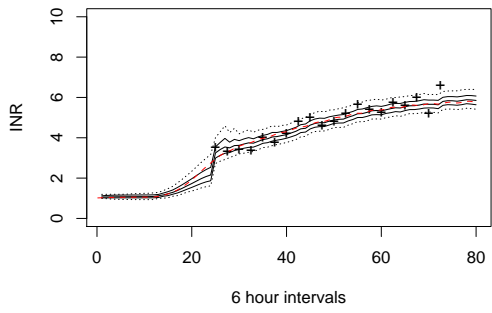


Figure 6.43

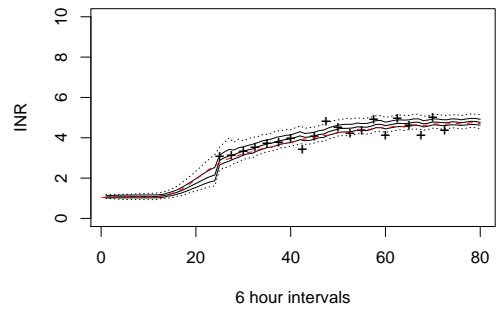


Figure 6.44

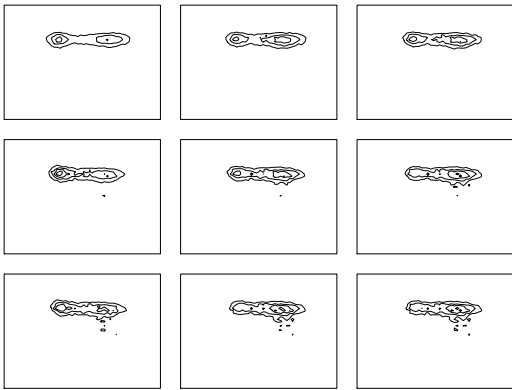
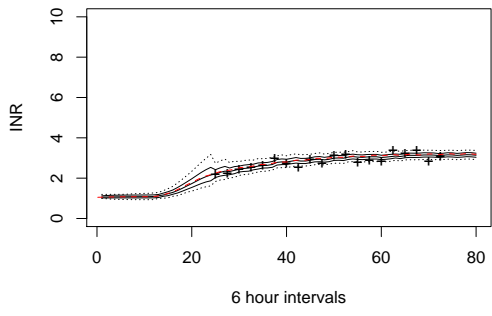


Figure 6.45

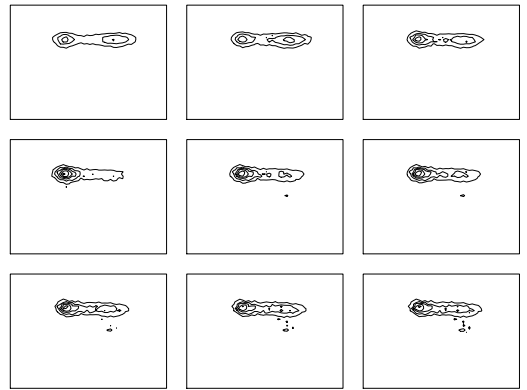
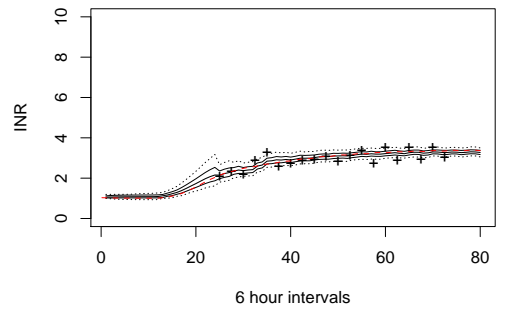


Figure 6.46

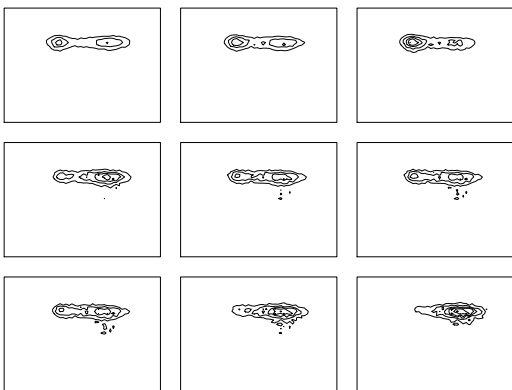
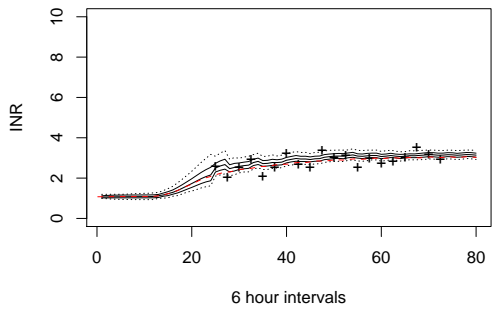


Figure 6.47

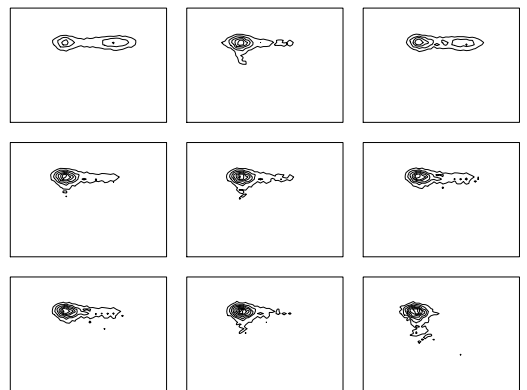
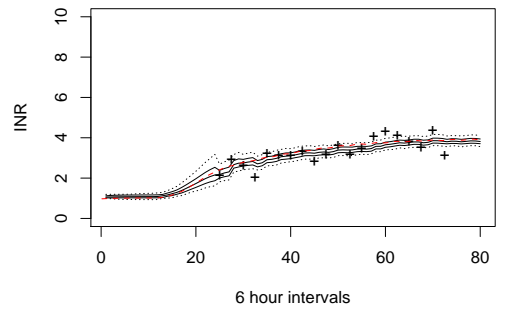


Figure 6.48

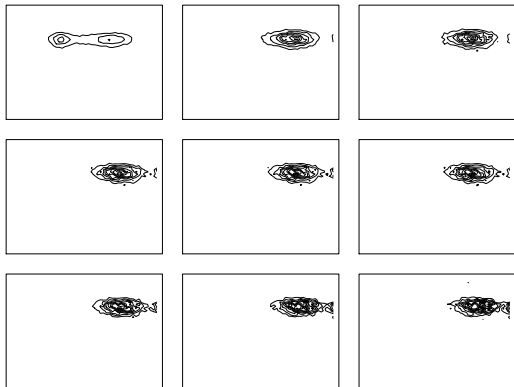
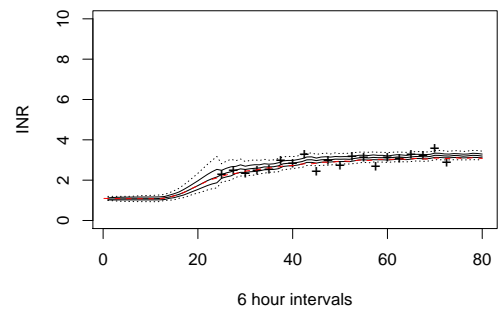
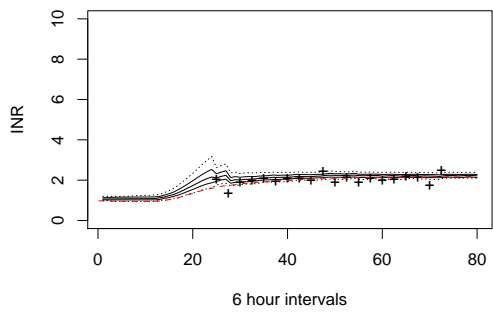


Figure 6.49

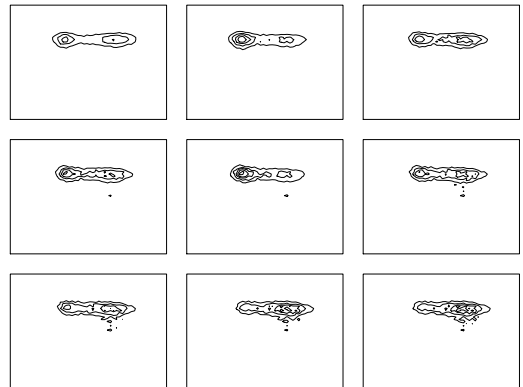


Figure 6.50