**Basic compartment models and non-linear mixed-effects modeling in R**

Tim Cardilin, tim.cardilin@fcc.chalmers.se

**Preparation**

Very basic knowledge of R is assumed. To prepare, make sure to have a recent version of R installed alongside (preferably) RStudio. Install the package ‘nlmixr2’ by following the instructions on the website: [Nonlinear Mixed Effects Models in Population PK/PD • nlmixr2](https://nlmixr2.org/) Note that nlmixr2 requires that Rtools has already been installed (see [RTools: Toolchains for building R and R packages from source on Windows (r-project.org)](https://cran.r-project.org/bin/windows/Rtools/)).

**Introduction**

This workshop centers around mathematical modeling of time series data. Generally, a time series is a sequence of measurements of a given quantity over time (e.g., a person’s blood pressure measured over several days; the price of a particular stock option over time).

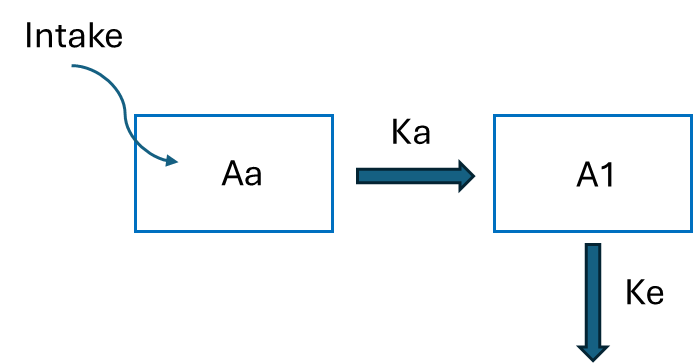
In nutrition, and in this workshop, we shall consider time series that describe the amount or concentration of a particular nutrient (molecule) upon ingestion, i.e., data governed by the processes of absorption, distribution, metabolism, and elimination. For example, we may have measured the amount of caffeine in a person’s blood after drinking a cup of coffee.

Constructing a mathematical model of such data would allow us to estimate the amount of caffeine at other time points. The model can also be used to simulate what would happen if we had a larger or smaller cup of coffee, or if we two hours after the first cup of coffee decided to drink another one.

The exercises below aim to familiarize the reader with a few classical models that are frequently employed to describe the concentration of an ingested nutrient over time in the body (blood). This is followed by a few exercises centered around model calibration or parameter estimation.

**Exercises**

We shall consider what are known as compartment models, which schematically consist of a set of boxes or compartments of well mixed material. The simplest such model is shown in the figure below.



The model consists of two compartments, Aa and A1, frequently representing the amount of a given molecule in the gut and blood, respectively. Upon intake at time t=0, the molecule is first placed in the gut from where it is absorbed into the blood with rate Ka. Once in the blood, the molecule is eliminated from the body with rate Ke. Mathematically, we describe this model using two ordinary differential equations:

The plasma concentration, C1, can be obtained by dividing the amount A1 with the distribution volume V1 as

We shall solve/simulate such systems using the package rxODE2 in R (and later estimate model parameters using the package nlmixr2).

To specify a model in rxODE2 we need two things:

1) an ‘ini’ object containing a set of model parameters and their (initial) values

2) a ‘model’ object which defines the model equations (i.e. the differential equations and any secondary equations).

To simulate the model we also need an event table (‘ev’) which defines the sampling times and the dosing/intake schedule. (If there is only a single dose at time t=0, then you can instead choose to define this in the initial conditions of the model. However, this will not work if you use the solved systems with ‘linCmt()’, see below.)

**Exercise 1:** Simulate the above model, and plot the concentration in blood, using the package ‘rxode2’ in R for the parameter values Ka=0.1 (1/hour), Ke=0.5 (1/hour), V1 = 4.0 (L), and Intake=1000 (ug). R code for how to do this is given below. Try increasing/decreasing the value of each model parameter, one at a time. What happens to the trajectory C1?

library(rxode2)

library(ggplot2)

mod <- function() {

# ini defines the model parameter values and their values

ini({

KA=0.1 # Absorption rate (1/hour)

KE=0.5 # Plasma clearance (L/hour)

V1=4.0 # Plasma volume (L)

Intake=1000 # of a given molecule (ug)

})

# model defines the model equations (incl. secondary equations)

model({

C1 <- A1/V1 # plasma concentration (ug/L)

d/dt(Aa) <- -KA\*Aa # turnover in depot/absorption compartment (ug/hour)

d/dt(A1) <- KA\*Aa - KE\*A1 # turnover in central/plasma compartment (ug/hour)

Aa(0) = Intake # of the modelled molecule (ug)

})

}

model <- mod() # create the ui object (can also use `rxode2(mod)`)

ev <- et(seq(from = 0, to = 48, by = 0.1)) # Add sampling over 0-48 hours in steps of 0.1 hours

sol <- rxSolve(mod,ev)

plot(sol, C1, log="y") + ylab("Plasma conc. (ug/mL")

**Comment:** See also [Single Subject ODE solving -- differences from multiple subject • rxode2 (nlmixr2.github.io)](https://nlmixr2.github.io/rxode2/articles/rxode2-single-subject.html)

We can add additional doses by changing the event table ‘ev’. For example, the code segment below adds 5 additional doses of 1000 ug to the compartment Aa, starting at t=24 hours, with a dosing interval of 24 hours (we also extend the sampling to 6 days):

ev <- et(amt=1000, nbr.doses=5, start.time=24, dosing.interval=24, cmt='Aa') %>% et(seq(from = 0, to = 24\*6, by = 0.1))

If you are not used to working with differential equations, some common models (including the ones we look at in this workshop) have automated solutions using rxode2 and nlmixr2. This means that you can replace ‘model’ by the line

C1 <- linCmt(KA,KE,V1,K12,K21)

When you use this syntax rxode2 will try to guess the model structure based on the name of the parameters. It therefore becomes vital that one uses typical parameter names. A list of acceptable choices can be found at. Note that you will not

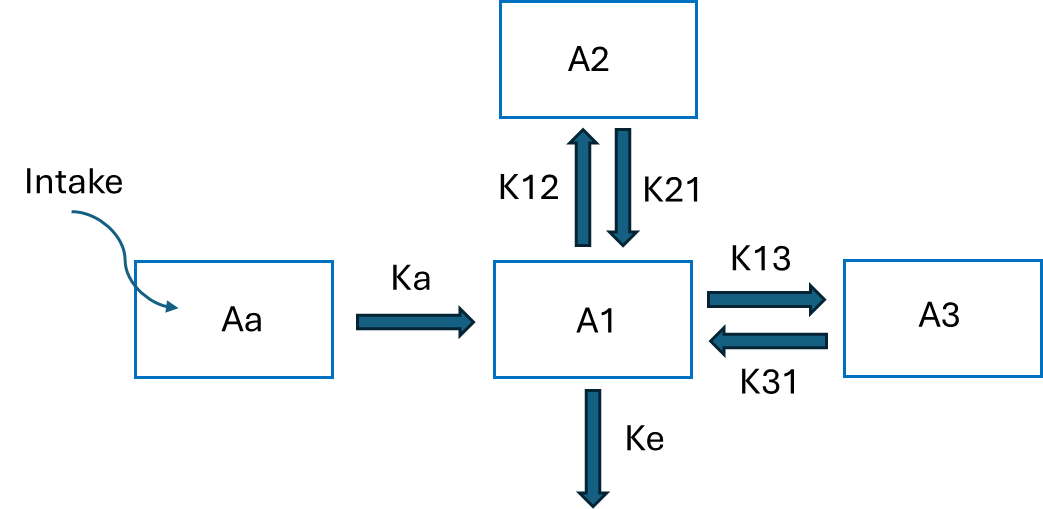
**Exercise 2:** Run the model in exercise, but change the intake to 1500 ug, which should be given three times at t=0, 24, 48. Feel free to play around with repeated dosing.

We next consider the slightly more complex compartment model (with an additional compartment) given by the differential equations:

where A2 denotes the amount of the molecule in (an unspecified) tissue. This model introduces two new parameters, K12 and K21, which correspond to the distribution of the molecule from blood to tissue and vice versa.

**Exercise 3:** First, draw a schematic figure of this model (similar to the one provided for the model in exercise 1). Then repeat exercises 1 and 2 using this new model. You can use the default parameters Ka = 2.7 (1/hour), Ke = 0.5 (1/hour), V1 = 10 (L), K12 =0.1 (1/hour), K21 = 0.2 (1/hour) and Intake = 1000 (ug). How does this model differ (quantitatively) from the previous model?

**Exercise 4:** Another model is schematically given in the figure below. Implement the model using rxode2, either by writing down the corresponding system of differential equations, or by using ‘linCmt()’. Compare with the previous models. As default parameters you can use K13 =1, K31 = 1.



Thus far, we have simulated time series for a single individual. If we instead consider a population of individuals, we expect some model parameters to vary between individuals, e.g., the elimination rate Ke. To describe this, we assume that Ke (in this case) is fixed to a value, rather it is drawn from specified probability distribution. In rxode2 we can write

ini({

TKe <- 1 # Typical value

eta.Ke ~ 0.1 # variability in Ke

})

which defines a typical Ke (called TKe) which is the typical (or median) value in the population, and a normally distributed eta.Ke (here with variance 0.1 and default mean value 0). Note that we use ‘~’ to indicate that eta.Ke is a random variable. In the model specification the elimination rate Ke becomes

model({

KE <- TKe\*exp(eta.Ke)

})

This will ensure that the parameter Ke is lognormally distributed with median value TKe. Whenever the model is solved/simulated, a new eta.Ke is drawn and a new Ke is calculated and used to solve the system of differential equations. **Note:** the lognormal distribution commonly used, one reason being that it ensures positivity of Ke.

**Exercise 5:** Choose the model from exercise 1 and simulate a population of individuals, choosing one or multiple parameters to vary within the population. To simulate e.g., 20 individuals you can use nSub=20 and write

sol <- rxSolve(mod, ev, nSub=20)

Realistically, models and measurements will not be perfect – they will contain measurement noise (or unexplained error). The two most common error models are the additive and proportional error models. An additive error is assumed to be normally distributed and independent of the measured quantity, whereas a proportional error is assumed to be proportional to the measured quantity. These can be implemented by making slight changes to ‘ini’ and ‘model’:

In the ‘ini’ object we can e.g. write:

prop.err= 0.1 # or add.err = 1

Then, in the ‘model’ we write:

C1 ~ prop(prop.err) # or C1 ~ add(add.err)

Finally, we simulate the model as usual,

sol <- rxSolve(mod, ev, nSub=100)

However, this time the simulated measurements (with added noise) will be located in the ‘sim’ column of the output table ‘sol’, so make sure that you plot the right thing.

**Exercise 6:** Continue from exercise 5 by adding additive or proportional noise, then simulate and plot the population. If you choose to simulate a large population, plotting all individual curves may look quite messy. Instead you can choose to plot only certain percentiles with ‘confint’ e.g.

s <- confint(sol, "sim", level=0.95)

plot(s)

Finally, we shall focus on calibrating models to simulated data, using the package ‘nlmixr2’ (see also [Nonlinear Mixed Effects Models in Population PK/PD • nlmixr2](https://nlmixr2.org/) ).

First, we need to understand how a dataset (consisting of time series data from multiple individuals) should be structured. At minimum, the following three columns are required: ID, TIME, DV.

Here ‘ID’ is used to differentiate between individuals. Thus ID=1 for the first individual in the population, ID=2 for the next one, etc. Next, TIME simply specifies the time point for a given measurement (e.g, we have a measured the blood concentration at t=5 hours after intake). DV stands for Dependent Variable which simply specified the value of the measured quantity (e.g., the blood concentration of molecule X was measured to be 10 ug/L).

An example of a simple dataset is given below

|  |  |  |
| --- | --- | --- |
| ID | TIME | DV |
| 1 | 0 | 100 |
| 1 | 1 | 90 |
| 1 | 2 | 80 |
| 2 | 0 | 95 |
| 2 | 1 | 90 |
| 2 | 2 | 88 |
| 3 | 0 | 102 |
| 3 | 1 | 97 |
| 3 | 2 | 79 |

This dataset consists of measurements for three individuals with IDs 1,2,3, with measurements at times 0,1,2. The value of the measured quantity is given by the column DV.

Assuming that data are in the right format, estimating the model parameters is straightforward using the command nlmixr2, e.g.

fit <- nlmixr2(mod, data1, est="focei")

This will use the data in the table ‘data1’ to estimate the parameters in the model ‘mod’. Moreover, the model must – as before – include an ‘ini’ object with initial estimates (guesses) for the model parameters (including any variability via eta parameters and model error) and an ‘model’ object which defines the model equations (including an error model). In other words, ‘mod’ must be in the same format as in exercise 6!

If we then simply type ‘fit’ this will give us a bunch of information about the model fit including the estimated parameter values and their uncertainties (in terms of relative standard error, RSE). We can also look at the individual fits alongside a number of diagnostic plots using plot(fit). Thi

In the above example we write est=”focei” to specify which estimation method should be used (in this case the First-Order Conditional Estimation method with Interaction). Nlmixr2 supports “focei” and “saem”. We recommend focei for smaller models with variability only in a few parameterers, whereas saem (which is a stochastic method) is much faster for larger models or models with variability in most (if not all) parameters.

**Exercise 7:** Use the model and data from exercise 6 to re-estimate the model parameters. Try changing the initial estimates (guesses) to something slightly different than the true values. In order to use the simulated dataset, you will need to rename the columns (to ID, TIME, and DV) using e.g., ‘colnames’.

**Note:** If you cannot get this to work, you can try copy+paste of the code below to see an example.

mod4 <- function() {

# ini defines the model parameter values and their values

ini({

# central

KA=2.7 # Absorption rate (1/hour)

TKE=0.5 # Typical elimination rate (1/hour)

eta.KE~0.1 # variability in elimination

V1=10 # Plasma volume (L)

# peripheral

K12=0.1 # transfer rate from plasma to peripheral (1/hour)

K21=0.2 # transfer rate from peripheral to plasma (1/hour)

Intake= fixed(1000) #fixed means do not estimate (ug)

prop.err= 0.1 # or add.err = 10

})

# model defines the model equations (incl. secondary equations)

model({

KE <- TKE\*exp(eta.KE)

C1 <- A1/V1 # plasma concentration (ug/L)

d/dt(Aa) <- -KA\*Aa # turnover in depot/absorption compartment (ug/hour)

d/dt(A1) <- KA\*Aa - KE\*A1 - K12\*A1 + K21\*A2 # plasma turnover (ug/hour)

d/dt(A2) <- K12\*A1 - K21\*A2 # turnover in peripheral compartment (ug/hour)

Aa(0) = Intake # of the modelled molecule (ug)

C1 ~ prop(prop.err) # or C1 ~ add(add.err)

})

}

ev<- et(seq(from = 0, to = 48, by = 1), cmt = "C1")

sol4 <- rxSolve(mod4, ev, nSub=20)

plot(sol4, sim, log="y") + ylab("Cp (ug/mL)") + scale\_y\_log10(limits =c(0.1,100))

#s <- confint(sol4, "sim")

#plot(s)

data <- sol4

colnames(data)[1] <- "ID" # rename first column to ID

colnames(data)[6] <- "DV" # rename 6th (in this example) column to DV

fit <- nlmixr2(mod4, data, est="focei")

fit

plot(fit)

**Exercise 8:** Try to find a good model for the example data (excel file to be found in the same file as this one). You can assume a singe intake of 10 mg at time t=0.

**Exercise 9:** Try to simulate data from different models, with different levels of variability and measurement noise, and then estimate the parameters. Generally, if you include a lot of data (meaning frequent sampling times, many individuals, etc.) and if you have low variability and measurement noise, it will be much easier to estimate the parameters accurately.

**Comment:** If you want to estimate a model with repeated dosing/intake, this must be integrated into the dataset. In addition to the columns ID, TIME, DV, you will also need columns AMT, EVID, and CMT which correspond to the amount, the “event ID” and the compartment that shall receive the input. You should set EVID = 0 for rows that correspond to a measurement, and EVID = 1 for a dosing/intake event. In the example below, we have augmented the previous dataset with intake into the compartment “Aa” at times t=0 and t=1. The first row indicates an intake of amount 1000 for the individual with ID 1 at time t=0. The third row indicates another intake of amount 1000 for the same individual at time t=1. Similarly, rows have been included to indicate intake for IDs 2 and 3. Note that EVID is set to 0 for measurements and 1 for intake/dosing.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | TIME | DV | AMT | EVID | CMT |
| 1 | 0 |  | 1000 | 1 | Aa |
| 1 | 0 | 100 |  | 0 | C1 |
| 1 | 1 |  | 1000 | 1 | Aa |
| 1 | 1 | 90 |  | 0 | C1 |
| 1 | 2 | 80 |  | 0 | C1 |
| 2 | 0 |  | 1000 | 1 | Aa |
| 2 | 0 | 95 |  | 0 | C1 |
| 2 | 1 |  | 1000 | 1 | Aa |
| 2 | 1 | 90 |  | 0 | C1 |
| 2 | 2 | 88 |  | 0 | C1 |
| 3 | 0 |  | 1000 | 1 | Aa |
| 3 | 0 | 102 |  | 0 | C1 |
| 3 | 1 |  | 1000 | 1 | Aa |
| 3 | 1 | 97 |  | 0 | C1 |
| 3 | 2 | 79 |  | 0 | C1 |