mrgsolve: Simulate from ODE-Based Models

Get Started Package Vignette

Kyle T. Baron

2023-01-04

mrgsolve is an R package maintained under the auspices of Metrum Research Group that facilitates simulation from models based on systems of ordinary differential equations (ODE) that are typically employed for understanding pharmacokinetics, pharmacodynamics, and systems biology and pharmacology. mrgsolve consists of computer code written in the R and C++ languages, providing an interface to a C++ translation of the lsoda differential equation solver.

This vignette will show you how to get started using mrgsolve.

Contents

This vignette introduces the mrgsolve workflow. First, load the package along with any other helper packages we need for this vignette.

```
library(mrgsolve)
library(dplyr)
```

1 Get started

To get started with mrgsolve, try using the built in model library like this

```
mod <- modlib("pk1", delta = 0.1)

out <- mrgsim(mod, events = ev(amt = 100))

out
```

```
Model:  pk1
Dim:    242 x 5
Time:   0 to 24
ID:     1
    ID time     EV   CENT     CP
1:   1  0.0   0.00  0.000 0.0000
2:   1  0.0 100.00  0.000 0.0000
3:   1  0.1  90.48  9.492 0.4746
4:   1  0.2  81.87 18.034 0.9017
5:   1  0.3  74.08 25.715 1.2858
6:   1  0.4  67.03 32.619 1.6309
7:   1  0.5  60.65 38.819 1.9409
8:   1  0.6  54.88 44.383 2.2191
```

```
plot(out, "CP")
```

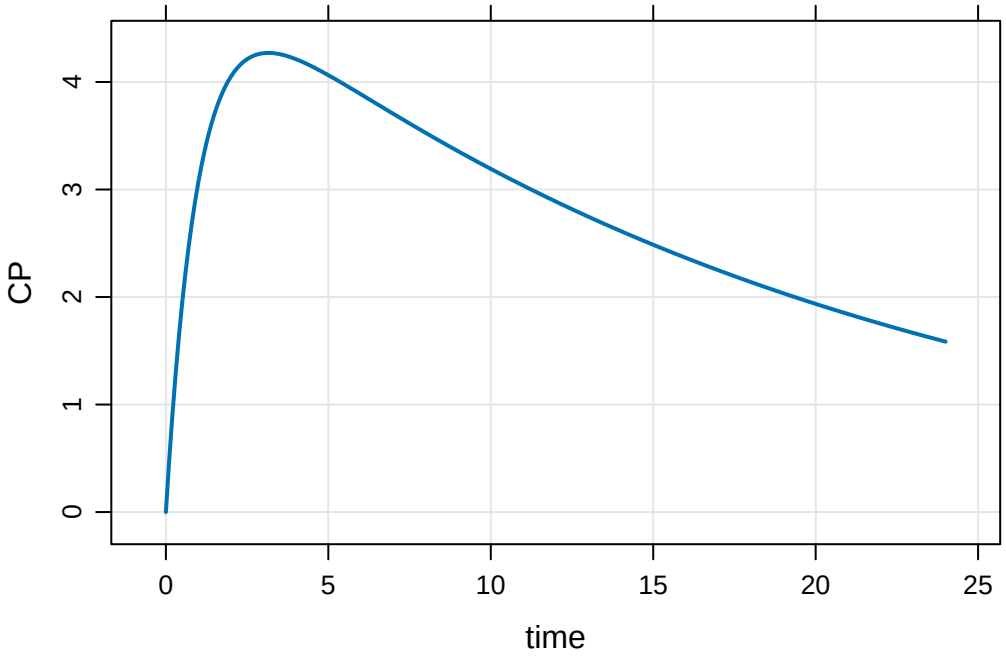


Figure 1: Simple simulation of a single dose

That was a really simple simulation where we used an event object to initiate a dose into a one-compartment model. See how the `plot()` method allows us to quickly visualize what happend in the simulation.

2 Event objects

Event objects help you implement dosing events with a lightweight, easy to compose syntax. You construct them with the `ev()` function. So I can make an object for a single 100 mg bolus dose

```
ev(amt = 100)
```

```
Events:
  time amt cmt evid
1    0 100   1    1
```

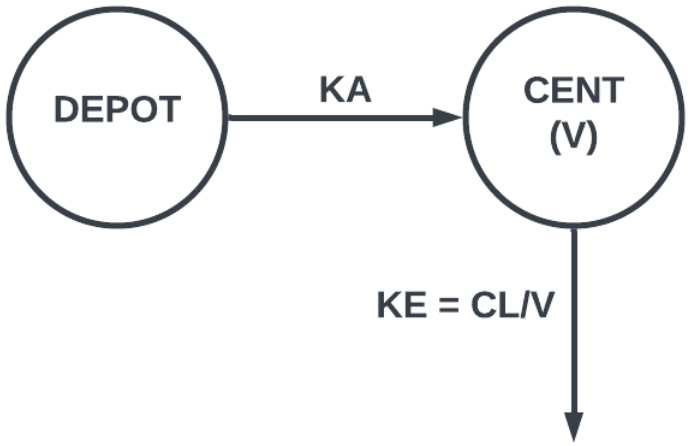
or we can code a series of intermittent infusions

```
ev(amt = 100, rate = 50, ii = 24, addl = 3)
```

```
Events:
  time amt rate ii addl cmt evid
1    0 100   50 24    3   1    1
```

3 Model Specification

In this chapter, we'll code up a very simple pharmacokinetic model. This is just starter material to get some concepts in place. We'll do more complicated stuff later (see **?@sec-specification-2**).



Parameters

- `CL`
- `V`
- `KA`

Compartments

- `CENT`
- `DEPOT`

3.1 Where to save your code?

- You can use any file name with any extension
- The extension you use may influence how your editor highlights and indents your code
- I use `.mod`

3.2 Model specification blocks

Model components are coded into blocks, which are delineated by a specific block syntax. You have a couple of options

NONMEM style

These start with `$` and then the block name (`$PK`)

Bracket style

Put the block name in brackets (`[ ERROR ]`)

Upper or lower case

You can use either:

- `$error`
- `[ pk ]`
- etc … they all work

3.2.1 Syntax

The "type" of code you write will vary from block to block. Sometimes it is an R-like syntax and sometimes it is `c++` code.

Don't worry if you don't know `c++`! We have taken a lot of the complexity out and with a handful of exceptions, the code should be pretty natural and similar to what you write in R.

We will show you more `c++` in chapters to come.

3.3 Base model blocks

3.3.1 Parameters

Use the `$PARAM` block header.

```
$PARAM
CL = 1, V = 20, KA = 1.1
```

Parameters have a **name** and a **value**, separated by `=`.

Parameter names can be upper or lower case. If you want punctuation, use underscore `_`.

Parameter values must *evaluate* to numeric.

Parameters can't be functions of other parameters *when writing the `$PARAM` block*. But there is a place where you can do this … we'll see this later on.

Multiple parameters can go on one line, but separate by comma.

3.3.2 Read it in with `mread()`

Point `mread()` at your model file to

- read in the model
- see if it compiles

```
mod <- mread("simple.mod")
```

```
Building simple_mod ... done.
```

We'll learn much more about `mread()` later on, but for now we need a way to check if our model coding is correct.

3.4 Compartments

```
$PARAM
CL = 1, V = 20, KA = 1.1

$CMT DEPOT CENT
```

Compartments are named

- Upper or lower case
- Punctuation use _

Order doesn't matter, except

- When putting dosing records into the data set
- You want to use the NONMEM-style coding

3.5 Differential equations

Now, we'll write ODE using `$DES` (or `$ODE`) block.

```
$PARAM
CL = 1, V = 20, KA = 1.1

$CMT DEPOT CENT

$DES
dxdt_DEPOT = -KA * DEPOT;
dxdt_CENT  =  KA * DEPOT - (CL/V)*CENT;
```

Left hand side is `dxdt_<compartment name>`.

Right hand side can reference

- Compartments
- Parameters
- Other quantities derived in `$DES` or `$PK`
- Other internal variables

Unlike `$PARAM` and `$CMT`, this is `c++` code

- Any valid `c++` code is allowed
- Each line (statement) should end in semi-colon `;`

3.6 Derived outputs

Like NONMEM, derived can be calculated in the `$ERROR` block.

```
$PARAM
CL = 1, V = 20, KA = 1.1

$CMT DEPOT CENT

$DES
dxdt_DEPOT = -KA * DEPOT;
dxdt_CENT  =  KA * DEPOT - (CL/V)*CENT;

$ERROR
double CP = CENT/V;
```

Like `$DES`, this block must be valid `c++` code.

Here we have created a new variable called `CP`, which is the amount in the central compartment divided by the central volume of distribution.

When we create a new variable, we must declare its `type`. Use `double` for a floating point number.

3.7 Capture outputs into the simulated data

`mrgsolve` has a `$CAPTURE` block that works like NONMEM's `$TABLE`. Just list the names you want copied into the output.

```
$PARAM
CL = 1, V = 20, KA = 1.1

$CMT DEPOT CENT

$DES
dxdt_DEPOT = -KA * DEPOT;
dxdt_CENT  =  KA * DEPOT - (CL/V)*CENT;

$ERROR
double CP = CENT/V;

$CAPTURE CP
```

Rather than putting stuff in `$CAPTURE`, try declaring with type `capture`

```
$ERROR
capture CP = CENT/V;
```

`capture` is identical to type `double`, but tells `mrgsolve` to include this item in the simulated output.

A little-use feature is renaming items in `$CAPTURE`

```
$ERROR
double DV = CENT/V;

$CAPTURE CP = DV
```