Regimen simulation with RxODE

July 22, 2020

Wenping Wang
On behalf of the nlmixr team

Outline

- RxODE basics
- Regimen simulation
- Population simulation
- “It’s Shiny!”
- Advanced topics

RxODE is pharmacometric simulation software as an open-source R package

- Written by Wenping Wang and Matt Fidler, available on CRAN[1] and GitHub[2], and described in a tutorial in CPT:PSP[3] and with online documentation[4]
- Simulation of ODEs was already possible in R (using deSolve), but was slow and virtually impossible to code with flexible dosing history
- RxODE allows fully flexible dosing history
- RxODE has rapid execution due to compilation in C
- Stable and mature software for Windows, OS X, Linux
- Requires external compilers (provided by Rtools on Windows)

[1] CRAN: https://cran.r-project.org/web/packages/RxODE

[2] GitHub: https://github.com/nlmixrdevelopment/RxODE

[3] Wang W et al. CPT:PSP (2016) 5, 3–10.

[4] RxODE packagedown: https://nlmixrdevelopment.github.io/RxODE

RxODE is an R package that facilitates simulations of PK/PD with flexible dosing

load RxODE	library(RxODE)
Compile ODEs	m1 <- RxODE({ C2 = centr/V2 d/dt(depot) =-KA*depot d/dt(centr) = KA*depot - CL*C2 })
system parameters	theta <- c(KA=.294, CL=18.6, V2=40.2)
dosing & sampling	ev <- eventTable() ev$add.dosing(dose=10000, nbr.doses=5) ev$add.sampling(0:240)
simulation	x <- solve(m1, theta, ev)

RxODE syntax

Specifying a model with ordinary differential equations (ODE)

- ODEs are specified in assignments
- Derivative written as d/dt(var_name).
- Derived variables are allowed.
- All referenced, undefined variables are assumed to be input parameters.
- Input parameters must be specified as a named vector. The order of the input parameter vector is unimportant; names of the input parameter vector must be a superset of the parameters in ODEs.
- Optional attributes of compartments can be specified.

Compile the ODE model and set the parameter values

```
library(RxODE)

## set up the system of differential equations (ODEs)
m1 <- RxODE({
  C2 = centr/V2
  C3 = peri/V3
  d/dt(depot) =-KA*depot
  d/dt(centr) = KA*depot - CL*C2 - Q*C2 + Q*C3
  d/dt(peri)  =                   Q*C2 - Q*C3
  d/dt(eff)   = Kin - Kout*(1-C2/(EC50+C2))*eff
  eff(0) = Kin/Kout
})

## provide the parameter values to be simulated:
theta <-
  c(KA=2.94E-01,
    CL=1.86E+01, V2=4.02E+01,     # central
    Q=1.05E+01, V3=2.97E+02,      # peripheral
    Kin=1, Kout=1, EC50=200)      # effects
```

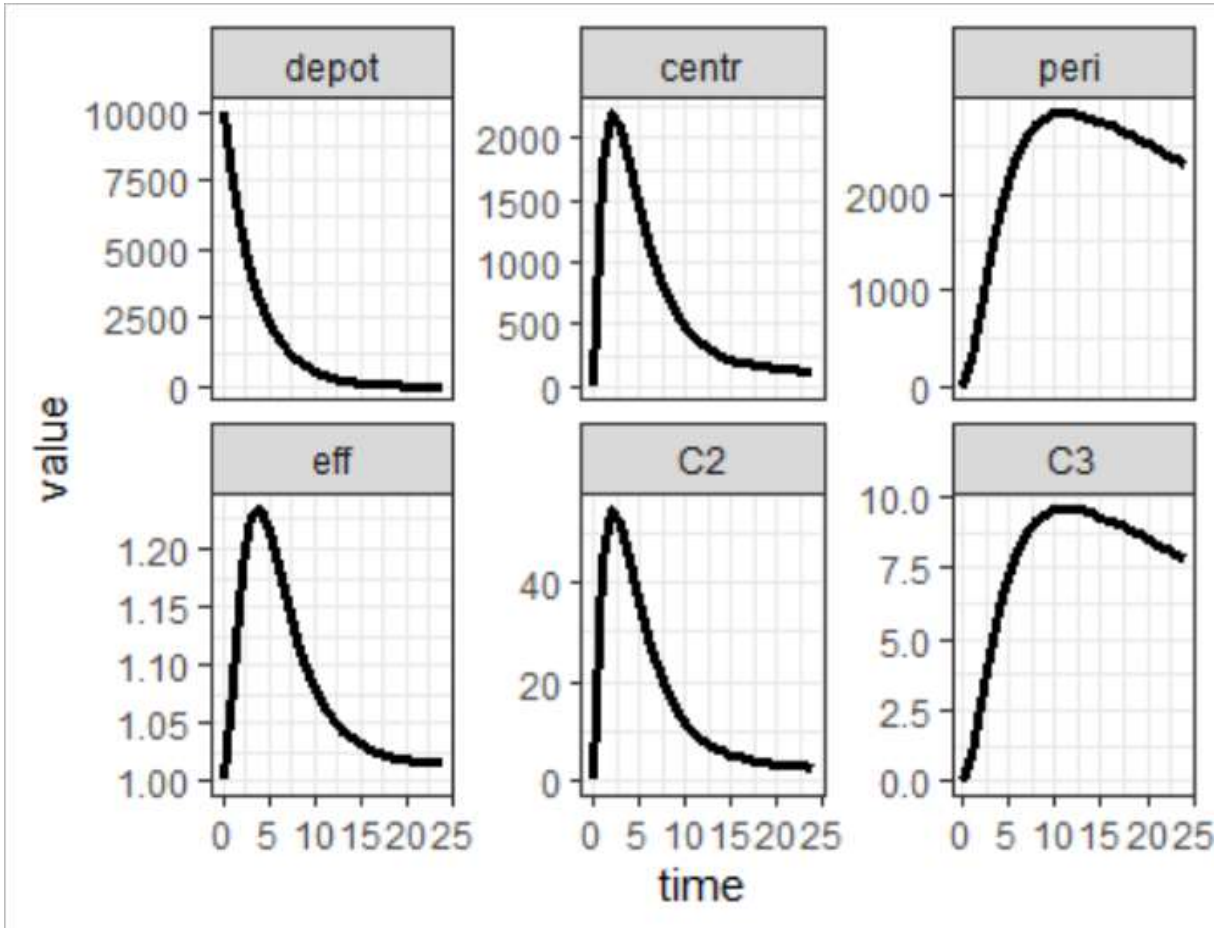
Create an eventTable that defines the doses and the sampling times

```
1  library(RxODE)
2
3  ## set up the system of differential equations (ODEs)
4  m1 <- RxODE({
5    C2 = centr/V2
6    C3 = peri/V3
7    d/dt(depot) =-KA*depot
8    d/dt(centr) = KA*depot - CL*C2 - Q*C2 + Q*C3
9    d/dt(peri)  =                    Q*C2 - Q*C3
10   d/dt(eff)  = Kin - Kout*(1-C2/(EC50+C2))*eff
11   eff(0) = Kin/Kout
12 })
13
14 ## provide the parameter values to be simulated:
15 theta <-
16   c(KA=2.94E-01,
17     CL=1.86E+01, V2=4.02E+01,     # central
18     Q=1.05E+01, V3=2.97E+02,      # peripheral
19     Kin=1, Kout=1, EC50=200)      # effects
20
21 ## create an empty event table that stores both dosing and sampling information
22 ev <- eventTable()
23
24 ## add a dose to the event table:
25 ev$add.dosing(dose=10000, nbr.doses=1)
26
27 ## add time points to the event table where concentrations will be simulated
28 ev$add.sampling(0:24)
```

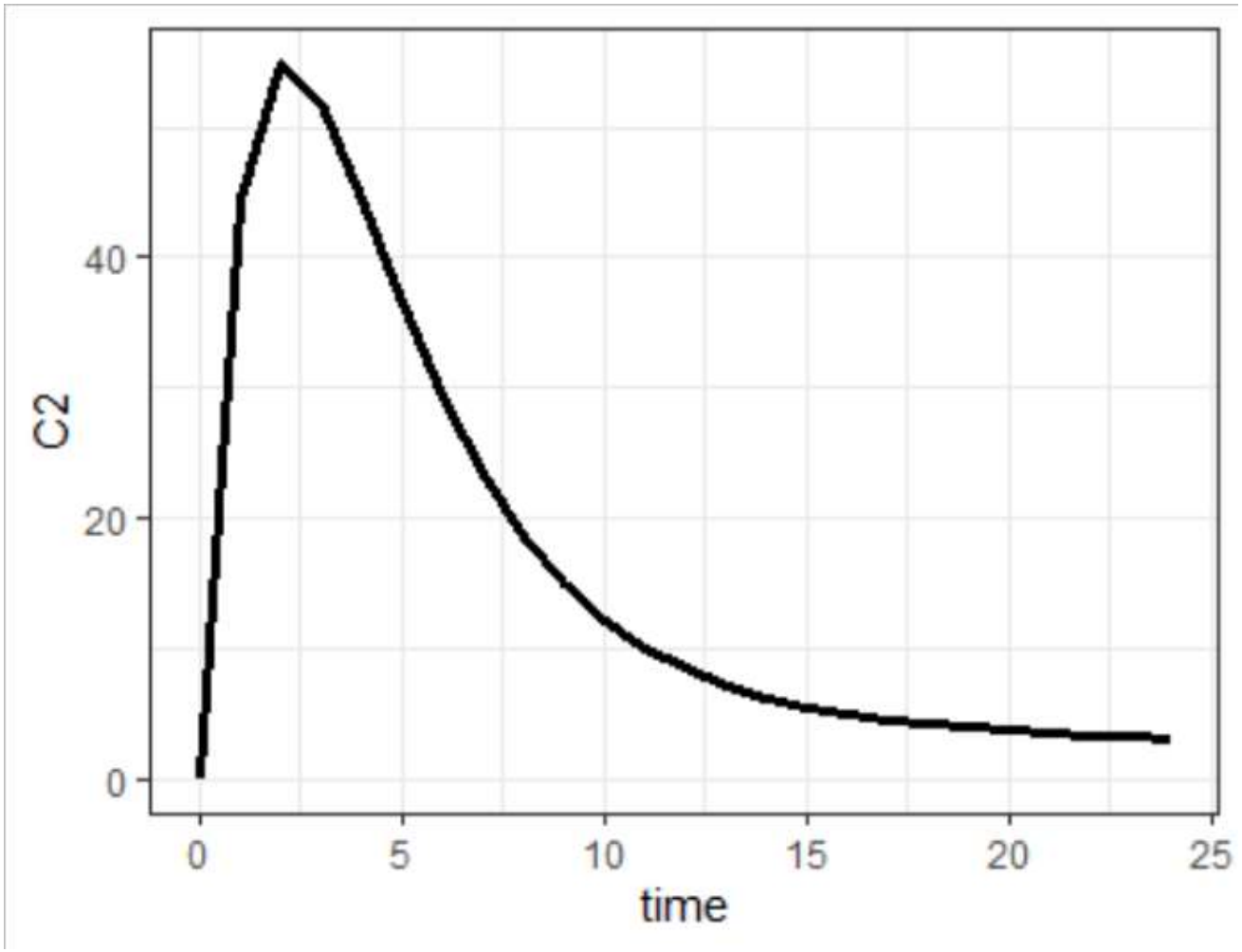
Run the model and plot the results

```
library(RxODE)

## set up the system of differential equations (ODEs)
m1 <- RxODE({
  C2 = centr/V2
  C3 = peri/V3
  d/dt(depot) =-KA*depot
  d/dt(centr) = KA*depot - CL*C2 - Q*C2 + Q*C3
  d/dt(peri)  =                    Q*C2 - Q*C3
  d/dt(eff)  = Kin - Kout*(1-C2/(EC50+C2))*eff
  eff(0) = Kin/Kout
})

## provide the parameter values to be simulated:
theta <-
  c(KA=2.94E-01,
    CL=1.86E+01, V2=4.02E+01,    # central
    Q=1.05E+01, V3=2.97E+02,     # peripheral
    Kin=1, Kout=1, EC50=200)     # effects

## create an empty event table that stores both dosing and sampling information
ev <- eventTable()

## add a dose to the event table:
ev$add.dosing(dose=10000, nbr.doses=1)

## add time points to the event table where concentrations will be simulated
ev$add.sampling(0:24)

## solve ODEs
x <- solve(m1, theta, ev)
## plot solution
plot(x, C2)
```

plot(x) shows the simulated time-course of all state-variables



plot(x, C2) shows the simulated time-course of a state-variable



Specify dosing and observation times with an EventTable()

An event table is a container that stores dosing and sampling times in chronological order.

- Incrementally add dosing records:

```
add.dosing(dose=10000, nbr.doses = 3)                    # loading doses
add.dosing(dose=5000, nbr.doses=14, dosing.interval=12)  # maintenance
```

- Incrementally add observation times:

```
add.sampling(time=c(1, 2, 4, 8, 16, 20, 24))             # sampling times
```

add.dosing() examined

`add.dosing = function(`	
`amt,`	dose amount
`nbr.doses,`	number of doses
`dosing.interval=24,`	dosing interval
`dosing.to=1,`	where to dose
`rate=NULL,`	infusion rate
`start.time=0`	dosing start time: offset
`)`	

add.dosing() can be incrementally called with additive effects.

solve() examined

`solve = function(`	
`  m1,`	RxODE model
`  theta,`	parameters
`  events,`	event table
`  stiff=TRUE,`	stiff system?
`  atol=1e-8,`	absolute tolerance
`  rtol=1e-8,`	relative tolerance
`  ...`	
)	

The output of the solve() function is a data.frame with time, the `amt` (amount) in **all** compartments across time and ALL derived variable(s) if any, each row corresponds to a time in the event table.

Regimen simulation

Population simulation

RxODE has built-in capability of population simulation.

```
12  # system parameters
13  nsub=100
14  theta.all <-
15     cbind(KA=2.94E-01, CL=1.86E+01*exp(rnorm(nsub,0,.1)),
16       V2=4.02E+01, Q=1.05E+01, V3=2.97E+02,
17       Kin=1, Kout=1, EC50=200)
18
19  # dosing & sampling
20  ev <- eventTable()
21  ev$add.dosing(dose=10000, nbr.doses=5, dosing.interval=24)
22  ev$add.sampling(0:120)
23
24  # simu
25  x <- solve(m1, theta.all, ev)
26  plot(x, C2)
```

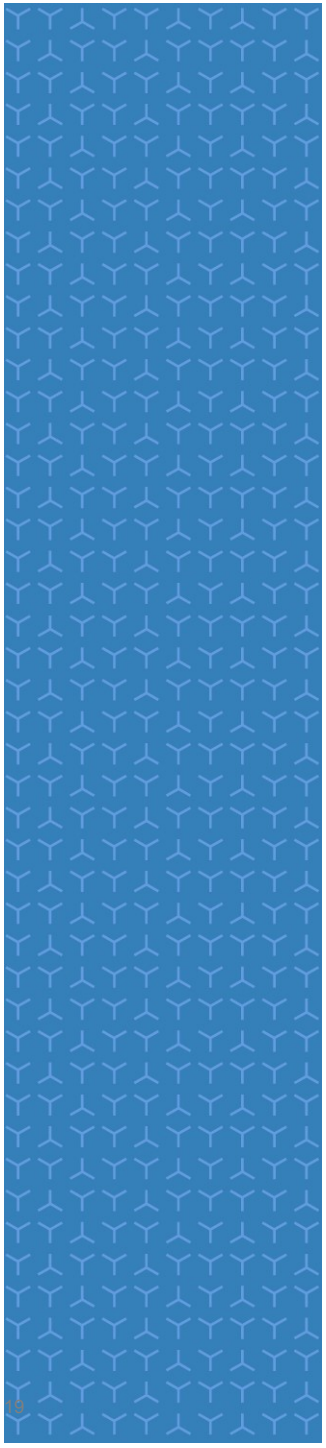
Shiny interface

“It’s Shiny”

Two interfaces:

- rxShiny()
- genShinyApp.template()

```
library(RxODE)
library(shiny)
genShinyApp.template(appDir = "myapp",
  ODE.config =
  list(
    ode = "C2 = centr/V2;
      C3 = peri/V3;
      d/dt(depot) =-KA*depot;
      d/dt(centr) = KA*depot - CL*C2 - Q*C2 + Q*C3;
      d/dt(peri)  =                    Q*C2 - Q*C3;",
    params = c(KA = 0.294, CL = 18.6, V2 = 40.2, Q = 10.5, V3 = 297),
    inits = c(depot = 0, centr = 0, peri = 0),
    method = "lsoda",
    atol = 1e-08, rtol = 1e-06)
)
runApp("myapp")
```



Advanced topics

Optional attributes of a compartment

`bioavailability`	f(depot)
`lag time`	lag(depot)
`modeled rate`	rate(depot)
`modeled duration`	dur(depot)
`initial value`	depot(0)

Simultaneous 0-order & 1-order absorption

```
m2 <- RxODE({
  C2 = centr/V2
  C3 = peri/V3
  d/dt(depot) =-KA*depot
  d/dt(centr) = KA*depot - CL*C2 - Q*C2 + Q*C3
  d/dt(peri)  =                   Q*C2 - Q*C3
  f(depot) = 1-F2
  f(centr) = F2
  dur(centr) = D2
})
```

Homework

Homework

1. Understand and execute ex1.R and ex2.R
2. Generate a Shiny App using the genShinyApp.template() function.
   - Modify the auto-generated UI.R and server.R to plot C2
   - Modify the auto-generated server.R to perform population simulation
3. Modify ex6.R to simulate sequential 0-order and 1-order absorption.
4. (bonus). Evaluate the QE approximation of TMDD via RxODE using parameters on Slide 24.

Evaluate the quasi-equilibrium approximation in TMDD

Adding (1) + (4), we get (8); adding (3) + (4), we get (10)

$$dC/dt = \mathrm{In}(t) - k_{on}R \cdot C + k_{off}\mathrm{RC} - (k_{el} + k_{pt})C + k_{tp} \cdot A_T/V_c \quad (1)$$

$$dA_T/dt = k_{pt}CV_c - k_{tp} \cdot A_T \quad (2)$$

$$dR/dt = k_{syn} - k_{on}R \cdot C + k_{off}\mathrm{RC} - k_{deg}R \quad (3)$$

$$d\mathrm{RC}/dt = k_{on}R \cdot C - (k_{off} + k_{int})\mathrm{RC} \quad (4)$$

$$dC_{tot}/dt = \mathrm{In}(t) - k_{int}C_{tot} - (k_{el} + k_{pt} - k_{int})C + k_{tp} \cdot A_T/V_c \quad (8)$$

$$dA_T/dt = k_{pt}CV_c - k_{tp}A_T \quad (9)$$

$$dR_{tot}/dt = k_{syn} - (k_{int} - k_{deg})(C_{tot} - C) - k_{deg}R_{tot} \quad (10)$$

C in (8) – (10) is undefined. We use the QE approximation to define C.

QE assumption:

$$\frac{R \cdot C}{\mathrm{RC}} = \frac{k_{off}}{k_{on}} \equiv K_D \quad (6)$$

which leads to:

$$C = 1/2\left[(C_{tot} - R_{tot} - K_D) + \sqrt{(C_{tot} - R_{tot} - K_D)^2 + 4K_D C_{tot}}\right]$$

Mager DE, Krzyzanski W: Pharm Res (2005) 22(10): 1589-1596.

Evaluate the quasi-equilibrium approximation in TMDD

```
4  # full TMDD
5  m1 <- RxODE({
6      kel = CL/V
7      kpt = Q/V
8      ktp = Q/Vt
9      d/dt(C)      =              -(kel+kpt)*C +ktp*peri/V -kon*C*R +koff*RC
10     d/dt(peri)   =                   kpt*C*V -ktp*peri
11     d/dt(R)      = ksyn -kdeg*R                          -kon*C*R +koff*RC
12     d/dt(RC)     = -kint*RC                              +kon*C*R -koff*RC
13     R(0) = ksyn/kdeg
14 })
```

```
62 # system parameters
63 theta <- c(
64   CL=0.15, V=3, Q=.45, Vt=3,
65   kon=.1, koff=.1,
66   kint=0.04, ksyn=1, kdeg=0.2
67 )
68
69 ev <- eventTable()
70 ev$add.dosing(dose=35, nbr.doses=1)
71 ev$add.sampling(seq(0, 100, by=0.1))
```