Case Studies

Contents

PKPD models and others...

Clinical trial simulation

Modeling and simulation workflow

Gene expression

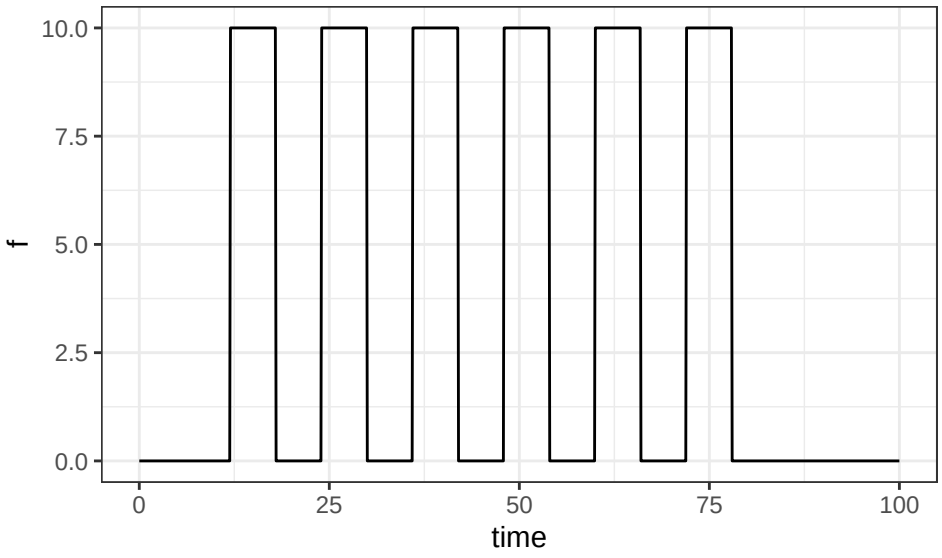
Tumor growth models

Step function

Regular steps - version 1

In this first version, the value of the step function is defined by the amount of the input pulses. Tlag defines the length of the step. Using p=-1 is equivalent to define a negative amount

```
stepModel1 <- inlineModel("
[LONGITUDINAL]
input = {tg}

PK:
depot(target=f)
depot(target=f, Tlag=tg, p=-1)

EQUATION:
ddt_f=0
")


res <- simulx(model=stepModel1,
              output=list(name='f', time=seq(0, 100, by=0.1)),
              treatment=list(amount=10, time=seq(12,78,by=12)),
              parameter=c(tg=6))

ggplot(res$f) + geom_line(aes(time,f))
```



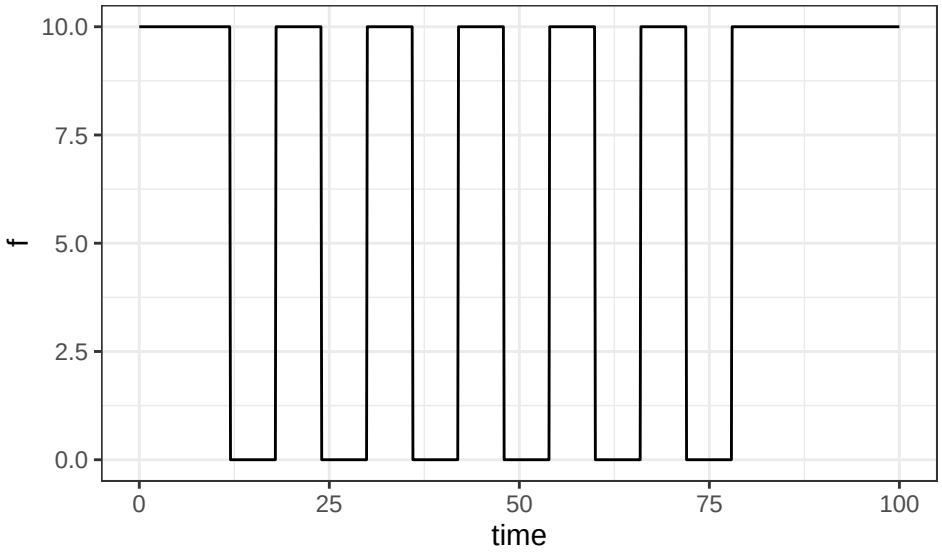
Regular steps - version 2

In this second version, the value of the step function is defined as an input parameter A. The amount of the input pulse amtDose is ignored since it is multiplied by p=A/amtDose.

Remark that we define f_0=A as initial condition in this example while the signal is a sequence of negative and positive pulses.

```
stepModel2 <- inlineModel("
[LONGITUDINAL]
```

```
input = {tg,A}

PK:
depot(target=f, p=-A/amtDose)
depot(target=f, Tlag=tg, p=A/amtDose)

EQUATION:
f_0 = A
ddt_f=0
")


res <- simulx(model=stepModel2,
              output=list(name='f', time=seq(0, 100, by=0.1)),
              treatment=list(amount=1, time=seq(12,78,by=12)),
              parameter=c(tg=6, A=10))

ggplot(res$f) + geom_line(aes(time,f))
```

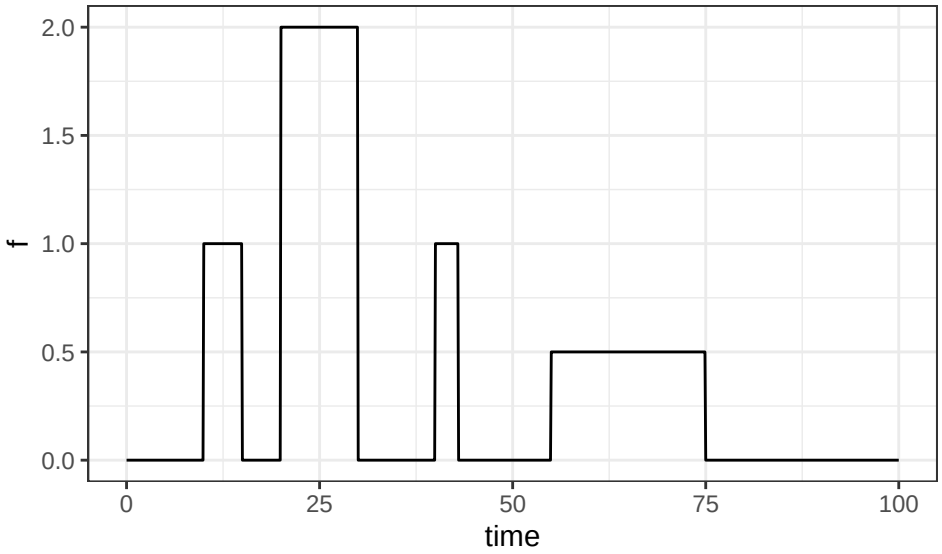


Irregular steps

In this example, the signal is a sequence of steps with irregular heights and lengths.

```
stepModel3 <- inlineModel("
[LONGITUDINAL]

PK:
depot(target=f, p=1,  type=1)
depot(target=f, p=-1, type=2)

EQUATION:
ddt_f=0
")

adm1 <- list(amount=c(1, 2, 1, 0.5), time=c(10, 20, 40, 55), type=1)
```

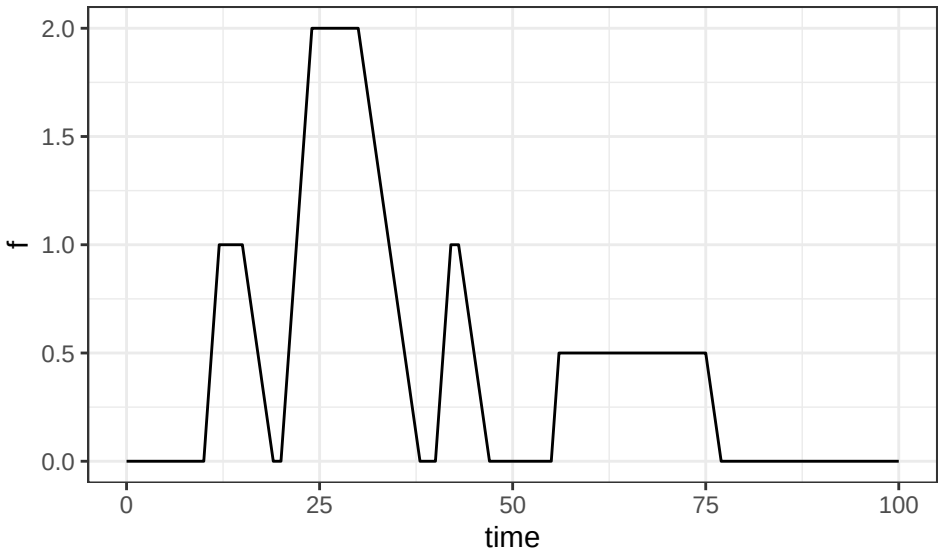
```
adm2 <- list(amount=c(1, 2, 1, 0.5), time=c(15, 30, 43, 75), type=2)
res <- simulx(model=stepModel3,
              output=list(name='f', time=seq(0, 100, by=0.1)),
              treatment=list(adm1, adm2))

ggplot(res$f) + geom_line(aes(time,f))
```



Step function with continuous transitions

Using infusions with constant rates lead to replace the Heavyside functions (sudden jumps) with continuous transitions.

```
adm1 <- list(amount=c(1, 2, 1, 0.5), time=c(10, 20, 40, 55), rate=0.5, type=1)
adm2 <- list(amount=c(1, 2, 1, 0.5), time=c(15, 30, 43, 75), rate=0.25, type=2)
res <- simulx(model=stepModel3,
              output=list(name='f', time=seq(0, 100, by=0.1)),
              treatment=list(adm1, adm2))

ggplot(res$f) + geom_line(aes(time,f))
```


2.0
1.5
1.0
0.5
0.0
f
0
25
50
75
100
time

Model for enterohepatic circulation

The structural model

We aim to implement and simulate a one compartment model with transit absorption compartments and enterohepatic circulation.

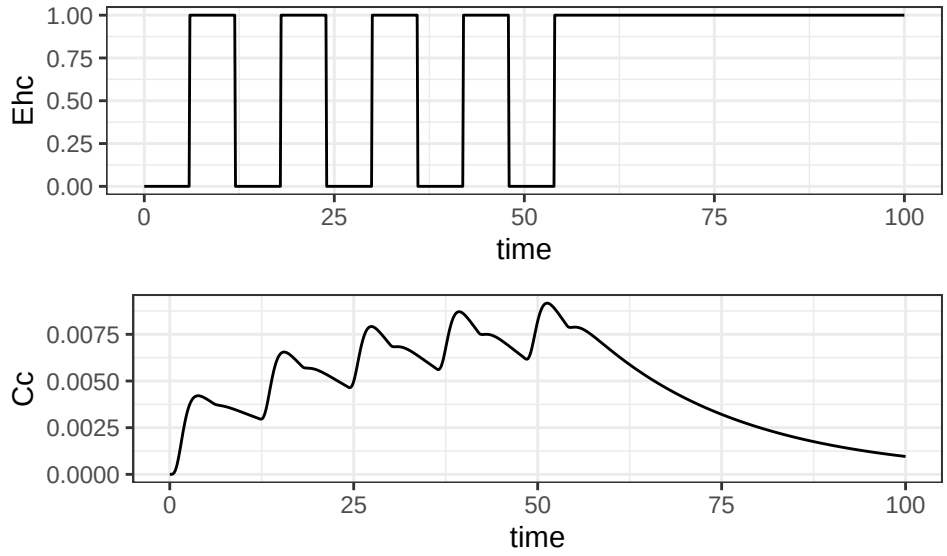
The step function is obtained by a sequence of "iv" administrations with successively negative and positive amounts. The step function jumps from 0 to 1 by setting `p= +1/amtDose`, and from 1 to 0 by setting `p= -1/amtDose`. here, `p` is the fraction of "dose" which is absorbed, then, the corresponding amount is `p * amtDose = +/- 1`

Let us implement first the model using PK macros.

```
model_macros <- inlineModel("
[LONGITUDINAL]
input = {Ntt, ka, kEhc, Fent, kb, Cl, V, tg}

PK:
compartment(cmt=1, amount=Ehc)
compartment(cmt=2, amount=Ak)
iv(cmt=1, p=-1/amtDose)
iv(cmt=1, Tlag=tg, p=1/amtDose)
oral(cmt=2, Mtt=Ntt/ka, Ktr=ka, ka)

EQUATION:
t0 = 0
Ehc_0 = 1
ddt_Ehc=0
ddt_Ak = Ehc*kEhc*Agb - ka*Ak
ddt_Acc = ka*Ak - Fent*kb*Acc - (1-Fent)*Cl/V*Acc
ddt_Agb = Fent*kb*Acc - Ehc*kEhc*Agb
Cc = Acc/V
")
```

We now use `simulx` for computing Ehc and Cc when multiple doses are administrated:

```
library(gridExtra)

p <- c(Cl=20, V=200, Ntt=3, kEhc=0.85, Fent=0.5, kb=0.04, ka=2.5, tg=6)
adm  <- list(amount=1, time=seq(0,50,by=12))
out  <- list(name = c('Ehc', 'Cc'), time = seq(0, 100, by=0.1))

res <- simulx(model=model_macros,
              output=out,
              treatment=list(adm),
              parameter=p)

pl1 <- ggplot(res$Ehc) + geom_line(aes(time,Ehc))
pl2 <- ggplot(res$Cc)  + geom_line(aes(x=time,y=Cc))
grid.arrange(pl1,pl2)
```



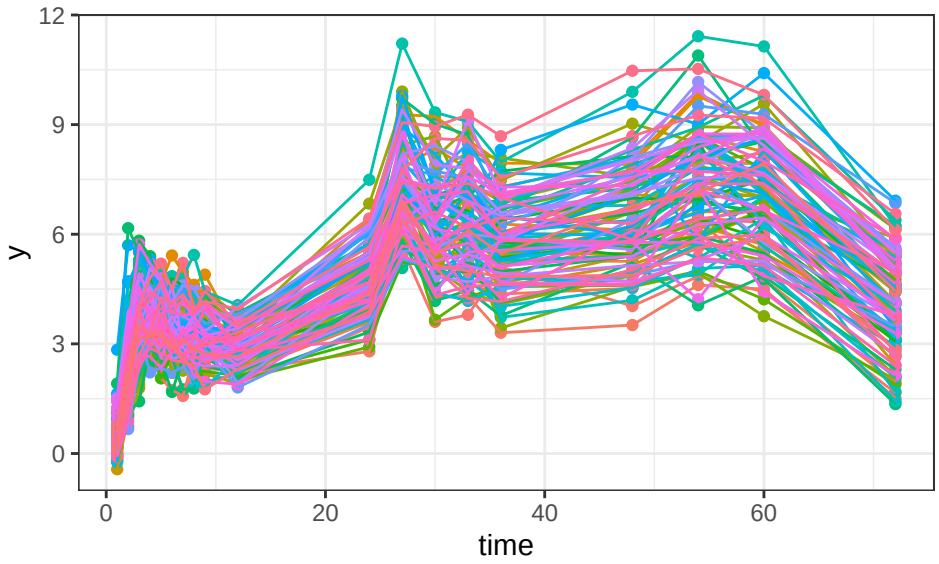
A Shiny application

```
adm.w <- list(
  tfd    = list(widget="slider", value=0,  min=0, max=24, step=2),
  nd     = list(widget="slider", value=4,  min=0, max=12, step=1),
  ii     = list(widget="slider", value=12, min=3, max=24, step=1),
  amount = list(widget="slider", value=1,  min=0, max=20, step=1)
)

p1 <- c(Cl=20, V=200, Ntt=3, ka=2.5)
p2 <- c(tg=6, kEhc=0.85, Fent=0.5, kb=0.04)
Ehc  <- list(name = 'Ehc', time = seq(0, 100, by=0.1))
Cc  <- list(name = 'Cc' , time = seq(0, 100, by=0.1))
out <- list(Ehc, Cc)

shiny.app <- shinymlx(model     = model_macros,
                      output    = list(Ehc, Cc),
                      treatment = adm.w,
                      parameter = list(p1,p2),
                      style     = "navbar1")
shiny::runApp(shiny.app)
```

Simulation of a clinical trial

```
enterohepatic_model <- inlineModel("
[LONGITUDINAL]
input = {Ntt, ka, kEhc, Fent, Cl, V, tg, a, b}

PK:
compartment(cmt=1, amount=Ehc)
compartment(cmt=2, amount=Ak)
iv(cmt=1, p=-1/amtDose)
iv(cmt=1, Tlag=tg, p=1/amtDose)
oral(cmt=2, Mtt=Ntt/ka, Ktr=ka, ka)
```

```
EQUATION:
kb = Cl/V
t0 = 0
Ehc_0 = 1
ddt_Ehc=0
ddt_Ak = Ehc*kEhc*Agb - ka*Ak
ddt_Acc = ka*Ak - kb*Acc
ddt_Agb = Fent*kb*Acc - Ehc*kEhc*Agb
Cc = Acc/V
g = a + b*Cc

DEFINITION:
y = {distribution=normal, prediction=Cc, sd=g}

[INDIVIDUAL]
input = {Cl_pop, omega_Cl, V_pop, omega_V, kEhc_pop, omega_kEhc,
Fent_pop, omega_Fent, ka_pop, omega_ka, tg_pop, omega_tg}

DEFINITION:
tg = {distribution=lognormal, prediction=tg_pop, sd=omega_tg}
Cl = {distribution=lognormal, prediction=Cl_pop, sd=omega_Cl}
V = {distribution=lognormal, prediction=V_pop, sd=omega_V}
ka = {distribution=lognormal, prediction=ka_pop, sd=omega_ka}
kEhc = {distribution=lognormal, prediction=kEhc_pop, sd=omega_kEhc}
Fent = {distribution=logitnormal, prediction=Fent_pop, sd=omega_Fent}

")

p <- c(Cl_pop   = 20,     omega_Cl = 0.2,
       V_pop    = 200,     omega_V = 0.2,
       kEhc_pop = 0.85, omega_kEhc = 0.2,
       Fent_pop = 0.5,  omega_Fent = 0.2,
       ka_pop   = 2.5,    omega_ka = 0.2,
       tg_pop   = 6,      omega_tg = 0.2,
       Ntt      = 3,
       a        = 0.3,
       b        = 0.02)

adm  <- list(amount=1000, time=seq(0,50,by=12))
Cc  <- list(name = 'Cc', time = seq(0, 100, by=0.1))
y   <- list(name = 'y',  time = c(1,2,3,4,5,6,7,8,9,12,24,27,30,33,36,48,54,60,72))

res <- simulx(model     = enterohepatic_model,
              output    = y,
              treatment = list(adm),
              group     = list(size=100, level='individual'),
              parameter = p)

ggplot() +  geom_point(data=res$y, aes(x=time, y=y, color=id)) +
  geom_line(data=res$y, aes(time,y, color=id)) +  theme(legend.position="none")
```




```
writeDatamlx(res, result.file="entero_data.csv")
```

Target-Mediated Drug Disposition (TMDD) Models

Introduction

We aim to implement several of the models used in this excellent tutorial on TMDD models:

> Dua P, Hawkins E, van der Graaf P. *A Tutorial on Target-Mediated Drug Disposition (TMDD) Models*. CPT: Pharmacometrics & Systems Pharmacology. 2015;4(6):324-337.

Following Dua *et al.*, "Target-mediated drug disposition (TMDD) is the phenomenon in which a drug binds with high affinity to its pharmacological target site (such as a receptor) to such an extent that this affects its pharmacokinetic characteristics.''

For more biological and mathematical details about the models used here, readers are referred to the paper of Dua *et al.* and the many references therein.

A one compartment model

The model

The one-compartment model was first proposed by Mager and Jusko in 2001:

> Mager, D. E., & Jusko, W. J. (2001). General pharmacokinetic model for drugs exhibiting target-mediated drug disposition. Journal of pharmacokinetics and pharmacodynamics, 28(6), 507-532. [PudMed]

Dua *et al.* propose a mathematical description of this model and a Berkeley Madonna code for this base TMDD model.

$$
\begin{aligned}
\frac{dL}{dt} &= -k_{eL}L - k_{on}LR + k_{off}P \quad ; \quad L(0) = 0 \\
\frac{dR}{dt} &= k_{in} - k_{out}R - k_{on}LR + k_{off}P \quad ; \quad R(0) = R_0 = k_{in}/k_{out} \\
\frac{dP}{dt} &= k_{on}LR - k_{off}P - k_{eP}P \quad ; \quad P(0) = 0
\end{aligned}
$$

This model was now implemented in `Mlxtran`:

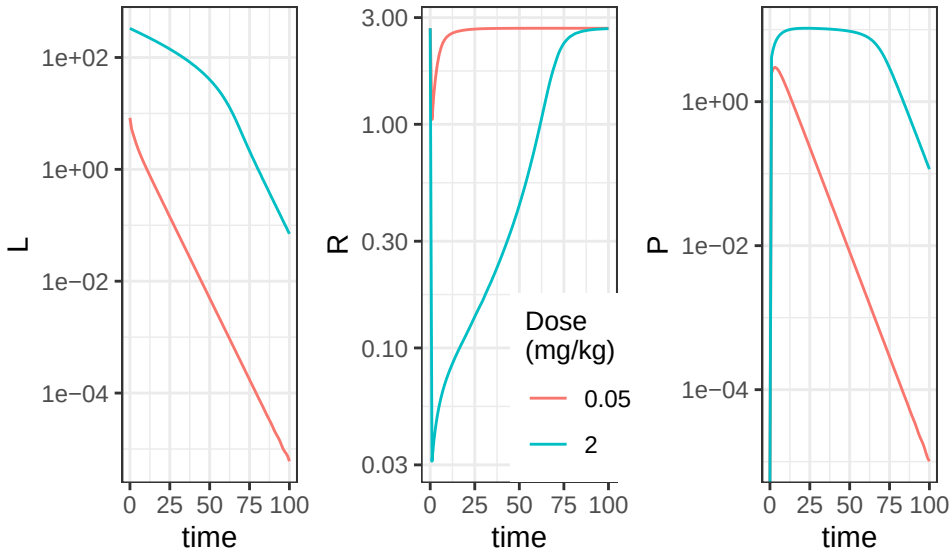
```
[LONGITUDINAL]
input = {kel, kep, kon, kout, koff, Vc, R0}

PK:
; Dosemg: Dose injected into central compartment (mg/kg)
; MWlig=150000; molecular weight of ligand (Da)
; (Dosemg*1e-3/MWlig)*1e9 ; Dose conversion (nmol/kg)

depot(target=L, p=100/15/Vc)  ; L is a concentration (nM)

EQUATION:
kin = kout*R0 ; Synthesis of receptor (nM/day)

;Initial Conditions
L_0 = 0
R_0 = R0
P_0 = 0

;Ordinary Differential Equations
ddt_L = -kel*L-kon*L*R+koff*P
```

```
ddt_R = kin-kout*R-kon*L*R+koff*P
ddt_P = kon*L*R-koff*P-kep*P
```

Comparison of 2 dosing regimens

We perform the same numerical experiment in order to show the nonlinear (dose- and time-dependent) behavior of the pharmacokinetics of the ligand.

```
library(gridExtra)

p <- c(
kel  = 0.024, # Elimination of ligand (1/day)
kep  = 0.201, # Elimination of complex (1/day)
kout = 0.823, # Elimination of receptor (1/day)
koff = 0.9,   # Dissociation rate (1/day)
kon  = 0.592, # Binding rate (1/(nMday))
R0   = 2.688,  # inital concentration of receptor  in central compartment (nM)
Vc   = 0.04  # volume of central compartment (L/kg)
)

## Dosing Regimen
Dosemg <- c(0.05, 2)
adm1   <- list(amount=Dosemg[1], time=0)
adm2   <- list(amount=Dosemg[2], time=0)
g1     <- list(treatment = adm1)
g2     <- list(treatment = adm2)

## Observations
tobs <- seq(0,100,by=1)
out  <- list(name = c('L', 'R', 'P'), time = tobs)

## Run `simulx`
res <- simulx(model    = "model/tmdd1.txt",
              parameter= p,
              output   = out,
              group    = list(g1, g2))

## Plot results
pl1 <- ggplot(data=res$L) + geom_line(aes(x=time, y=L, colour=group), size=0.5) +
  theme(legend.position="none") + scale_y_log10()
pl2 <- ggplot(data=res$R) + geom_line(aes(x=time, y=R, colour=group), size=0.5) +
  scale_colour_discrete(name="Dose\n(mg/kg)", breaks=c("1","2"),labels=Dosemg) +
   theme(legend.title=element_text(size=10), legend.position=c(0.8,0.2))  + scale_y_log10()
pl3 <- ggplot(data=res$P) + geom_line(aes(x=time, y=P, colour=group), size=0.5) +
   theme(legend.position="none") + scale_y_log10()

grid.arrange(pl1,pl2,pl3, nrow=1)
```



A Shiny application

```
adm <- list(
  tfd    = list(widget="slider", value=0,  min=0,  max=24, step=2),
  nd     = list(widget="slider", value=1,  min=1,  max=10, step=1),
  ii     = list(widget="slider", value=12, min=4,  max=24, step=2),
  amount = list(widget="slider", value=1,  min=.1, max=2,  step=.1)
)

out1 <- list(name = 'L', time = tobs)
out2 <- list(name = 'R', time = tobs)
out3 <- list(name = 'P', time = tobs)

shiny.app <- shinymlx(model     = "model/tmdd1.txt",
                      parameter = list(p[1:4],p[5:7]),
                      output    = list(out1, out2, out3),
                      treatment = adm,
                       style     = "navbar2",
                       title     = "TMDD - Model 1")

shiny::runApp(shiny.app)
```

A two compartments model

This model, first described by Mager and Jusko in 2001, assumes that only unbound drug can distribute into a peripheral tissue compartment, while other ligands remain in the central compartment.

Quasi Equilibrium (QE) model

> *Quasi-equilibrium pharmacokinetic model for drugs exhibiting target-mediated drug disposition.* Mager DE, Krzyzanski W, Pharm Res. 2005 Oct; 22(10):1589-96. PubMed

The QE model proposed by Mager and Krzyzanski assumes that equilibrium between the binding and dissociation of the complex has been achieved. See Dua *et al.* for more details.

Quasi-steady-state (QSS) model

Parameter values proposed by Dua *et al.*

> *Approximations of the target-mediated drug disposition model and identifiability of model parameters*. Gibiansky L, Gibiansky E, Kakkar T, Ma P, J Pharmacokinet Pharmacodyn. 2008 Oct; 35(5):573-91. PubMed

The QSS model assumes that the binding rate is balanced by the sum of the dissociation and internalization rates. See Dua *et al.* for more details.

Parameter values for denosumab

> *Population pharmacokinetic analysis of denosumab in patients with bone metastases from solid tumours.* Gibiansky L, Sutjandra L, Doshi S, Zheng J, Sohn W, Peterson MC, Jang GR, Chow AT, P?rez-Ruixo JJ Clin Pharmacokinet. 2012 Apr 1; 51(4):247-60. PubMed

Michaelis-Menten (MM) model

The MM model is derived from the Michaelis-Menten equation for enzyme kinetics, which relates reaction rate to concentration. See Dua *et al.* for more details.

Multiple targets TMDD model

> *Target-mediated drug disposition model for drugs that bind to more than one target.* Gibiansky L, Gibiansky E, J Pharmacokinet Pharmacodyn. 2010 Aug; 37(4):323-46. PubMed

Some drugs have the ability to bind to multiple targets in one cell; for example, they could bind to both cell membrane (M) and soluble (S) targets. See Dua *et al.* for more details.

FcRn mediated recycling

> *Pharmacokinetics of anti-hepcidin monoclonal antibody Ab 12B9m and hepcidin in cynomolgus monkeys.* Xiao JJ, Krzyzanski W, Wang YM, Li H, Rose MJ, Ma M, Wu Y, Hinkle B, Perez-Ruixo JJ, AAPS J. 2010 Dec; 12(4):646-57. PubMed

Xiao *et al.* extended the QE two-compartment model with a depot to include the process of FcRn-mediated endosomal recycling. See Dua *et al.* for more details.

Individualised therapies

Task

Individualised therapies are increasingly important for drug development. One reason for individualised therapies is concentration related toxicity and high between-patient variability of exposure levels. In such a case each patient can be monitored for the concentration levels and the drug dose will be adjusted when the individual concentration level exceeds a certain threshold. In this example we demonstrate how `simulx` is used to run clinical trial simulations with an adaptive dosing. We will simulate a PK/PD model with a decision rule to adjust the dose based on monitored concentration levels. In addition, we will include a decision rule for the efficacy as well to ensure that desired efficacy level will be maintained.

The PK model

The PK model is a one compartment model for oral administration with first order absorption and elimination.

Parameters ka, V and k are log-normally distributed

This PK model is implemented in a text file `cts1a.txt`:

```
[LONGITUDINAL]
input = {ka, V, k, a1}
EQUATION:
Cc = pkmodel(ka, V, k)
DEFINITION:
y1 = {distribution=lognormal, prediction=Cc, sd=a1}

[INDIVIDUAL]
input={ka_pop, omega_ka, V_pop, omega_V, k_pop, omega_k}
DEFINITION:
ka = {distribution=lognormal, prediction=ka_pop, sd=omega_ka}
V  = {distribution=lognormal, prediction=V_pop,  sd=omega_V}
k  = {distribution=lognormal, prediction=k_pop,  sd=omega_k}
```

Simulation of PK data

We can use `simulx` for simulating a clinical trial with 20 patients to whom a dose of 20mg is administrated orally every 12 hours starting at time 0.

We then want to compute for these 20 patients:

- the predicted concentration Cc every hour between times 0h and 440h
- the observed concentration y_1 every 24 hours between time 4h and 440h

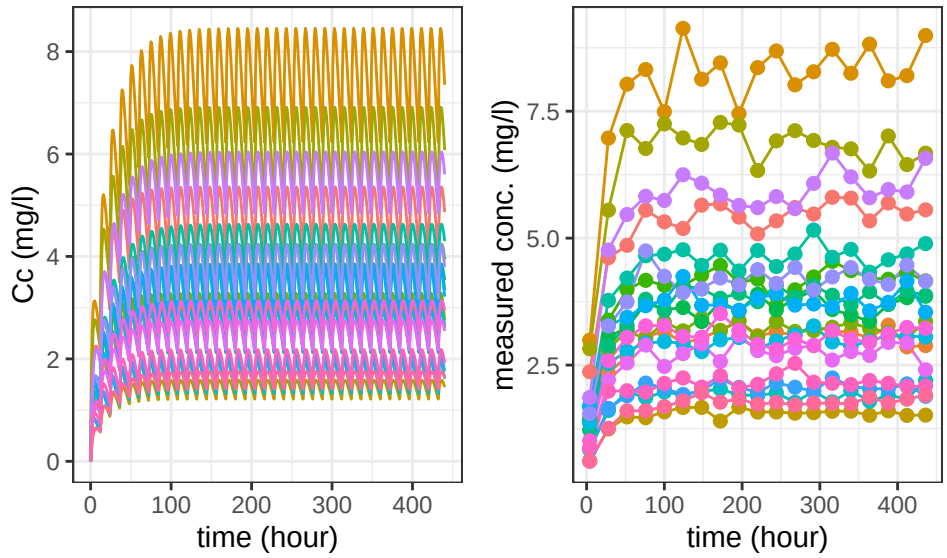
```
adm <- list(time=seq(0,to=440,by=12), amount=20)

ppk <- c(ka_pop=0.4,   V_pop=10,    k_pop=0.05,
         omega_ka=0.3, omega_V=0.5, omega_k=0.1,
         a1=0.05)

g   <- list(size=20, level='individual')

Cc <- list(name='Cc', time=seq(0,to=440,by=1))
y1  <- list(name='y1', time=seq(4,to=440,by=24))

s   <- list(seed=1234)

res1 <- simulx(model     = "model/cts1a.txt",
```

```
                parameter = ppk,
                treatment = adm,
                group     = g,
                output    = list(Cc, y1),
                settings  = s)
```

(we use seed = 1234 for reproducibility reason. Different results will be obtained with different values of the seed.)

We can now plot the predicted and observed concentrations:

```
library("gridExtra")
library(ggplot2)
theme_set(theme_bw())
plot1 <- ggplot(data=res1$Cc, aes(x=time, y=Cc, colour=id)) +
  geom_line(size=0.5) + xlab("time (hour)") + ylab("Cc (mg/l)") +
  theme(legend.position="none")
plot2 <- ggplot(data=res1$y1, aes(x=time, y=y1, colour=id)) +
  geom_line(size=0.5) + geom_point(size=2) +
  xlab("time (hour)") + ylab("measured conc. (mg/l)") +
  theme(legend.position="none")
grid.arrange(plot1, plot2, ncol=2)
```



Introducing a single decision rule

Assume now that, for safety reason, the concentration should remain below a given threshold of 5 mg/l.

The exposure, i.e. the "true'' concentration Cc, is above this threshold for several patients. Since Cc cannot be measured in practice, we can then decide to reduce the dose for those patients which measured concentration is above 5 mg/l.

We thus define an adaptive individual design as follows:

- Visits are scheduled every 24 hours, starting at time 124h.
- If, at a given visit, the observed concentration y_1 of a patient is greater than 5, then the next doses administrated to this patient will be reduced, by multiplying the current amount by a factor 0.75.

This decision rule is implemented as a R list:

```
r1 <- list(name='y1',
           time=seq(124,to=440,by=24),
           condition="y1>5",
           factor=0.75)
```

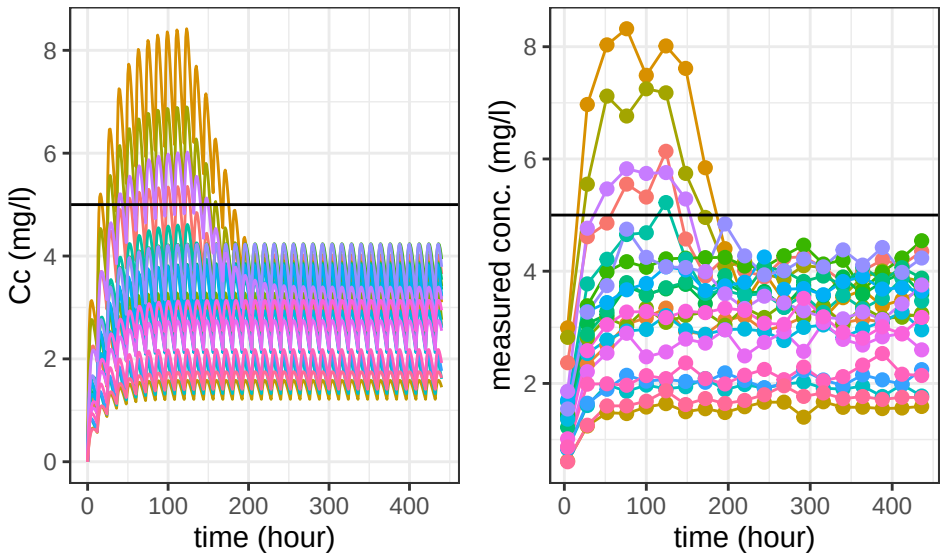
R function titration is a variant of `simulx` which now takes this rule into account.

Remark: function `titration.R` is not part of the `mlxR` package. This is just an example of what can be done: this function can be freely used/modified by anyone.

```
source("titration.R")
res2 <- titration(model     = "model/cts1a.txt",
                  parameter = ppk,
                  treatment = adm,
                  output    = list(Cc, y1),
                  rule      = r1,
                  group     = g,
                  settings  = s)
```

Outputs of `titration` and `simulx` have similar structure. We can therefore display the predicted and observed concentrations obtained with `titration` as we did previously with the output of `simulx`:

```
plot3 <- ggplot(data=res2$Cc, aes(x=time, y=Cc, colour=id))     +
  geom_line(size=0.5) + xlab("time (hour)") + ylab("Cc (mg/l)") +
  theme(legend.position="none")  + geom_hline(yintercept=5)
plot4 <- ggplot(data=res2$y1, aes(x=time, y=y1, colour=id))     +
  geom_line(size=0.5) + geom_point(size=2) +
  xlab("time (hour)") + ylab("measured conc. (mg/l)")           +
  theme(legend.position="none") + geom_hline(yintercept=5)
grid.arrange(plot3, plot4, ncol=2)
```



What was expected is now clearly observed: high concentrations tend to reduce after the first visit at time 124.

Combining several decision rules

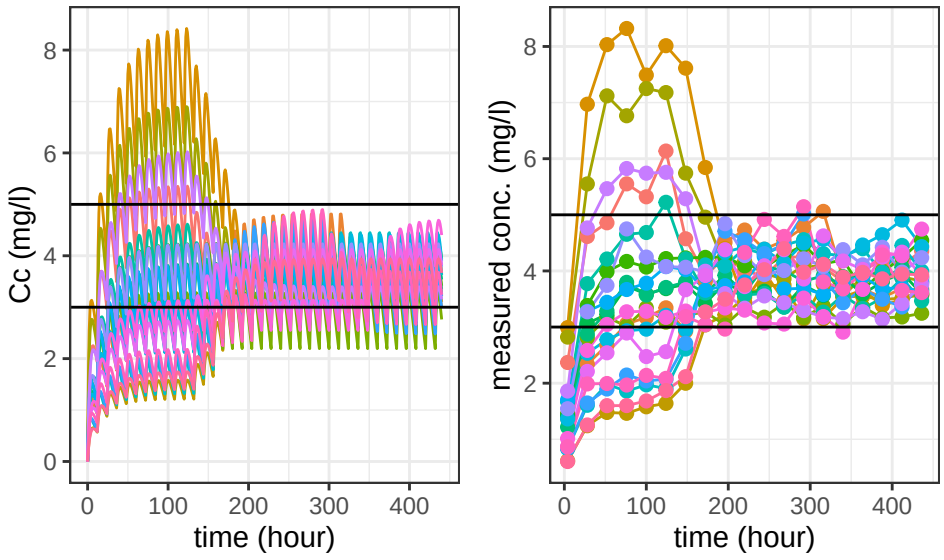
We can introduce several decision rules. Imagine for instance that efficacy requires a concentration above 3 mg/l. We can then decide, for example, to multiply the dose by a factor 1.5 when the observed concentration is below this value at a given visit.

We can then implement this new decision rule as a new list:

```
r2 <- list(name='y1',
           time=seq(124,to=440,by=24),
           condition="y1<3",
           factor=1.5)
```

We will now use `titration` combining the two decisions rules `r1` and `r2`:

```
res3 <- titration(model     = "model/cts1a.txt",
                  parameter = ppk,
                  treatment = adm,
                  output    = list(Cc, y1),
                  rule      = list(r1,r2),
                  group     = g,
                  settings  = s)

plot5 <- ggplot(data=res3$Cc, aes(x=time, y=Cc, colour=id))     +
  geom_line(size=0.5) + xlab("time (hour)") + ylab("Cc (mg/l)") +
  theme(legend.position="none") + geom_hline(yintercept=c(3,5))
plot6 <- ggplot(data=res3$y1, aes(x=time, y=y1, colour=id))     +
  geom_line(size=0.5) + geom_point(size=2) +
  xlab("time (hour)") + ylab("measured conc. (mg/l)")   +
  theme(legend.position="none") + geom_hline(yintercept=c(3,5))
grid.arrange(plot5, plot6, ncol=2)
```



Extension to PKPD data

Assume now that we have PK and PD data. In this example, the PD model is a Emax model.

The joint PKPD model is implemented in a new file `cts1b.txt`:

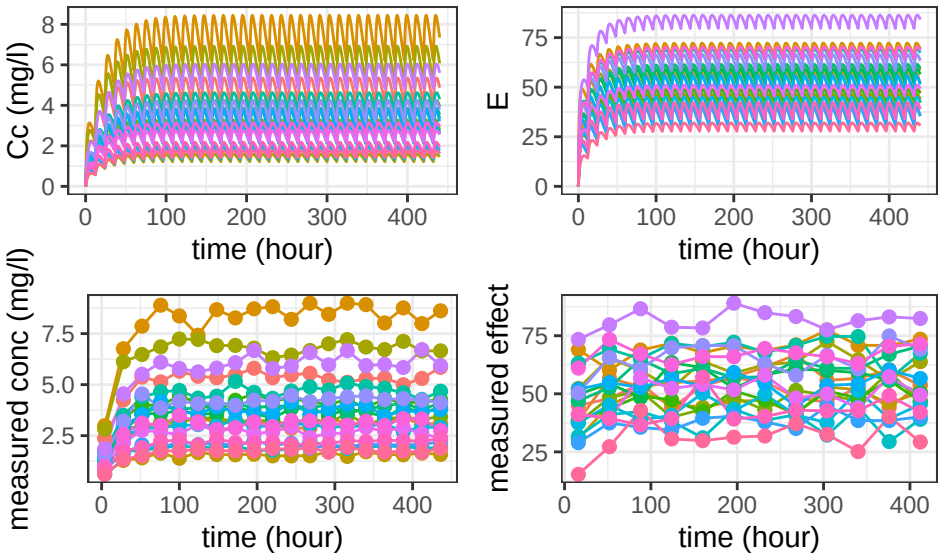
```
[LONGITUDINAL]
input = {ka, V, k, Emax, EC50, a1, a2}
EQUATION:
Cc = pkmodel(ka, V, k)
E  = Emax*Cc/(EC50+Cc)

DEFINITION:
y1 = {distribution=lognormal, prediction=Cc, sd=a1}
y2 = {distribution=normal, prediction=E,  sd=a2}

[INDIVIDUAL]
input={
ka_pop, omega_ka, V_pop, omega_V, k_pop, omega_k,
Emax_pop, omega_Emax, EC50_pop, omega_EC50
}
DEFINITION:
ka   = {distribution=lognormal, prediction=ka_pop,   sd=omega_ka}
V    = {distribution=lognormal, prediction=V_pop,    sd=omega_V}
k    = {distribution=lognormal, prediction=k_pop,    sd=omega_k}
Emax = {distribution=lognormal, prediction=Emax_pop, sd=omega_Emax}
EC50 = {distribution=lognormal, prediction=EC50_pop, sd=omega_EC50}
```

We need to provide the values of the population PD parameters for simulating PKPD data from this model.

```
ppd <- c(Emax_pop=100,   EC50_pop=3,
         omega_Emax=0.1, omega_EC50=0.2, a2=5)

E <-  list(name='E',  time=seq(0,to=440,by=1))
y2 <- list(name='y2', time=seq(16,to=440,by=36))

res4 <- simulx(model     = "model/cts1b.txt",
               parameter = c(ppk,ppd),
               treatment = adm,
               output    = list(Cc, E, y1, y2),
               group     = g,
               settings  = s)
```

We can now plot the predicted and observed PK and PD data for the 20 patients:

```
pl1=ggplot(data=res4$Cc, aes(x=time, y=Cc, colour=id))+geom_line(size=0.5)+
  xlab("time (hour)") + ylab("Cc (mg/l)") + theme(legend.position="none")
pl2=ggplot(data=res4$E, aes(x=time, y=E, colour=id))+geom_line(size=0.5)  +
  xlab("time (hour)") + ylab("E") + theme(legend.position="none")
pl3=ggplot(data=res4$y1, aes(x=time, y=y1, colour=id))+geom_line(size=0.5)+
  geom_point(size=2) + xlab("time (hour)") + ylab("measured conc (mg/l)") +
  theme(legend.position="none")
pl4=ggplot(data=res4$y2, aes(x=time, y=y2, colour=id))+geom_line(size=0.5)+
  geom_point(size=2) + xlab("time (hour)") + ylab("measured effect")      +
  theme(legend.position="none")

grid.arrange(pl1, pl2, pl3, pl4, ncol=2)
```

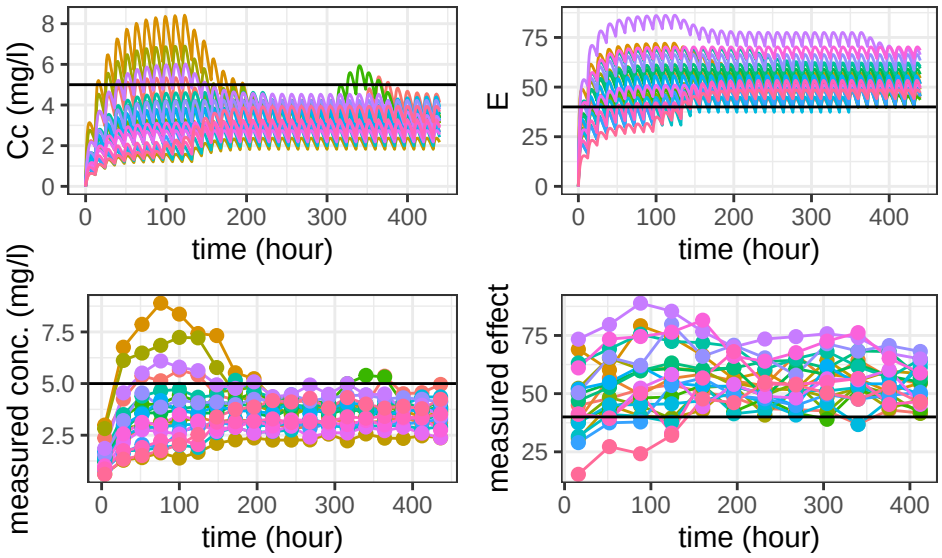


Efficacy can now be defined using the PD data. We may want an effect above 40 for instance. The decision rule is now based on the observed PD data y_2:

```
r3 <- list(name='y2',
           time=seq(88,to=440,by=36),
           condition="y2<40",
           factor=1.5)
```

`titration` is now used by combining rule `r1` (safety) and rule `r3` (efficacy):

```
res5 <- titration(model     = "model/cts1b.txt",
                  parameter = c(ppk,ppd),
                  treatment = adm,
                  output    = list(Cc, E, y1, y2),
                  rule      = list(r1,r3),
                  group     = g,
                  settings  = s)

pl5=ggplot(data=res5$Cc, aes(x=time, y=Cc, colour=id))+geom_line(size=0.5)+
  xlab("time (hour)") + ylab("Cc (mg/l)") + theme(legend.position="none") +
  geom_hline(yintercept=5)
pl6=ggplot(data=res5$E, aes(x=time, y=E, colour=id)) + geom_line(size=0.5)+
  xlab("time (hour)") + ylab("E") + theme(legend.position="none") +
  geom_hline(yintercept=40)
pl7=ggplot(data=res5$y1, aes(x=time, y=y1, colour=id))+geom_line(size=0.5)+
  geom_point(size=2)+xlab("time (hour)")+ylab("measured conc. (mg/l)")    +
  theme(legend.position="none") +  geom_hline(yintercept=5)
pl8=ggplot(data=res5$y2, aes(x=time, y=y2, colour=id))+geom_line(size=0.5)+
  geom_point(size=2) + xlab("time (hour)") + ylab("measured effect")      +
  theme(legend.position="none") + geom_hline(yintercept=40)

grid.arrange(pl5, pl6, pl7, pl8, ncol=2)
```



Remark: When both PK and PD data are simulated, the simulated PK data is different from those obtained when only PK data is simulated, even if the same seed is used. Indeed, neither the number of simulated individual parameters nor the number of simulated residual errors are the same.

Comparing the effect from different dose-regimen on survival in cancer patients: part I

Task

An optimal dose-regimen for the treatment of cancer is critical to maximise the effect while keeping the toxicity acceptable. PK/PD simulations are great way to explore all possible dose-regimen scenarios and to identify the one with the best efficacy-to-toxicity ratio. In this example we will focus on efficacy. We will compare and assess how four different dose-regimen will impact survival.

The example is structured in three parts. In the first part we simulate and compare the concentration, effect and survival for a single patient. The second part will include the between-patient variablity. The third part will investigate the full study design via Monte-Carlo clinical trial simulations and how the study size impacts our ability to retreive the drug effect on survival.

The PKPD model

For this example we will work with a PK/PD model where the parameters have been previously estimated and are known. The PK model is a one compartment model for oral administration with zero order absorption and linear elimination.

The PD is modeled in two parts. The first part describes the underlying disease process via a turnover model and the second part links the disease process with survival. The drug effect is modeled as an inhibition of the turnover model input.

The objective of this first part is to compare the predicted concentration, effect and survival. There is no observation and no inter-individual variability for the moment.

The PKPD model is implemented in the text file `model/cts2I.txt`:

```
[LONGITUDINAL]
input={Tk0,V,Cl,Imax,E0,IC50,kout,alpha,beta}

EQUATION:
Cc = pkmodel(Tk0, V, Cl)

E_0 = E0
kin = E0*kout
ddt_E = kin*(1-Imax*Cc/(Cc+IC50)) - kout*E

h = (alpha/1000)*exp(beta*E)
H_0 = 0
ddt_H = h
S = exp(-H)
```

Computing predictions

To run simulations with `simulx` the model parameter and the simulation design need to be specified in addtion to the structural model. Our study design has four treatment groups (defined as `groups` in `simulx`). Each group receive a dose every 12 hours, starting at time 0. For each of the group a different dose will be tested.

```
adm.time <- seq(0,200,by=12)
trt1 <- list(time=adm.time, amount=  0)
trt2 <- list(time=adm.time, amount= 25)
trt3 <- list(time=adm.time, amount= 50)
trt4 <- list(time=adm.time, amount=100)
```

```
g1   <- list(treatment = trt1)
g2   <- list(treatment = trt2)
g3   <- list(treatment = trt3)
g4   <- list(treatment = trt4)
```

The same PKPD parameters are used for the four groups.

```
ppk  <- c(Tk0   = 3,   V    = 10,  Cl   = 1)
ppd  <- c(Imax  = 0.8, E0   = 100, IC50 = 4, kout = 0.1)
ptte <- c(alpha = 0.5, beta = 0.02)
```

We specify the output for `simulx` by defining a list `output`. Here, this list tells `simulx` to compute the predicted concentration Cc, the predicted effect E and the predicted survival function S at every hour, starting at time 0:

```
f <- list(name = c('Cc','E','S'),
          time = seq(0,200,by=1))
```

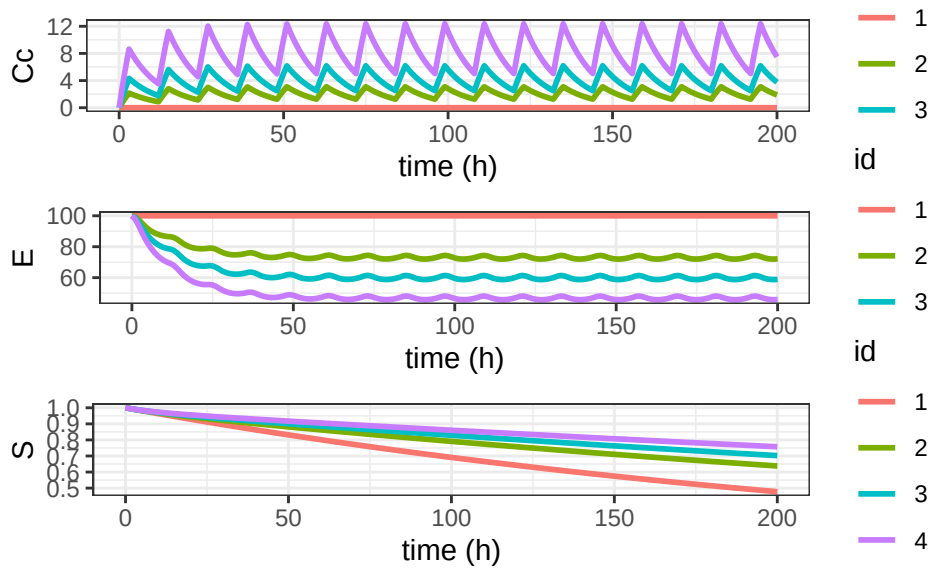
We can now run `simulx` with these input arguments

```
res <- simulx(model     = "model/cts2I.txt",
              group     = list(g1,g2,g3,g4),
              parameter = list(ppk, ppd, ptte),
              output    = f)
names(res)
```

```
## [1] "Cc"        "E"         "S"         "treatment"
```

and plot the results

```
library(gridExtra)

plot1 <- ggplot(data=res$Cc, aes(x=time, y=Cc, colour=id)) +
         geom_line(size=1) + xlab("time (h)") + ylab("Cc")
plot2 <- ggplot(data=res$E,  aes(x=time, y=E,  colour=id)) +
         geom_line(size=1) + xlab("time (h)") + ylab("E")
plot3 <- ggplot(data=res$S,  aes(x=time, y=S,  colour=id)) +
         geom_line(size=1) + xlab("time (h)") + ylab("S")
grid.arrange(plot1, plot2, plot3, ncol=1)
```



These simulation are for a single individual. To have more realisting predictions we will run simulation of a population PKPD model. We will see in part II how to compare these same quantities, but taking into account the inter-individual variability of the PKPD parameters.

Comparing the effect from different dose-regimen on survival in cancer patients: part II

Task

In the second part of this demonstration the between patient variability will be included to run population PKPD simulations. This will provide a more realistic prediction on the outcome among a large number of patients.

The PKPD model

The same PKPD model as in part I is used. To include the between-patient variablity the section `INDIVIDUAL` is added. In this section the distribution type and the distribution parameters of the PK parameters are sepcified.

Here log-normal distributions are used, except for $Imax$. Since $Imax$ can only take values between 0 and 1 a logit-normal distribution is applied.

This new joint PKPD model is implemented in the text file `model/cts2II.txt`:

```
[LONGITUDINAL]
input={Tk0,V,Cl,Imax,E0,IC50,kout,alpha,beta}

EQUATION:
C = pkmodel(Tk0, V, Cl)

E_0 = E0
kin = E0*kout
ddt_E = kin*(1-Imax*C/(C+IC50)) - kout*E

h = (alpha/1000)*exp(beta*E)
H_0 = 0
ddt_H = h
S = exp(-H)

[INDIVIDUAL]
input={Tk0_pop,V_pop,Cl_pop,Imax_pop,E0_pop,IC50_pop,kout_pop,alpha_pop,
       beta_pop, omega_Tk0,omega_V,omega_Cl,omega_Imax,omega_E0,omega_IC50,
       omega_kout,omega_alpha,omega_beta}

DEFINITION:
Tk0   = {distribution=lognormal,   prediction=Tk0_pop,  sd=omega_Tk0}
V     = {distribution=lognormal,   prediction=V_pop,    sd=omega_V}
Cl    = {distribution=lognormal,   prediction=Cl_pop,   sd=omega_Cl}
E0    = {distribution=lognormal,   prediction=E0_pop,   sd=omega_E0}
IC50  = {distribution=lognormal,   prediction=IC50_pop, sd=omega_IC50}
kout  = {distribution=lognormal,   prediction=kout_pop, sd=omega_kout}
Imax  = {distribution=logitnormal, prediction=Imax_pop, sd=omega_Imax}
alpha = {distribution=lognormal,   prediction=alpha_pop,sd=omega_alpha}
beta  = {distribution=lognormal,   prediction=beta_pop, sd=omega_beta}
```

Simulating inter individual variability (IIV)

The simulation here with `simulx` is analogous to the simulation in part I. One difference is that now we specify the number of patient to be used for the simulation. For each patient `simulx` will take a random sample from the parameter distributions and run a simulation. As a result we will obtain the distributions

for the concentration C, disease process E and survival S. In other words `simulx` produces by Monte Carlo simulation the predicted distributions of C, E and S. A large number of patients must then be simulated for each group in order to estimate correctly these distributions. Here, the size of the Monte Carlo is $N = 1000$ for each group.

The input for `simulx` to define four groups with four different doses and 1000 patients is:

```
N=400
adm.time <- seq(0,200,by=12)
trt1 <- list(time=adm.time, amount=  0)
trt2 <- list(time=adm.time, amount= 25)
trt3 <- list(time=adm.time, amount= 50)
trt4 <- list(time=adm.time, amount=100)
g1   <- list(treatment = trt1, size=N, level='individual')
g2   <- list(treatment = trt2, size=N, level='individual')
g3   <- list(treatment = trt3, size=N, level='individual')
g4   <- list(treatment = trt4, size=N, level='individual')
```

We use the same PKPD population parameters for the four groups.

```
pop.param   <- c(
  Tk0_pop   = 3,    omega_Tk0   = 0.2,
  V_pop     = 10,   omega_V     = 0.2,
  Cl_pop    = 1,    omega_Cl    = 0.2,
  Imax_pop  = 0.8,  omega_Imax  = 0.5,
  E0_pop    = 100,  omega_E0    = 0.1,
  IC50_pop  = 4,    omega_IC50  = 0.1,
  kout_pop  = 0.1,  omega_kout  = 0.1,
  alpha_pop = 0.5,  omega_alpha = 0.2,
  beta_pop  = 0.02, omega_beta  = 0
)
```

We specify the output for `simulx` by defining a list `output`. Here, this list tells `simulx` to compute the predicted concentration Cc, the predicted effect E and the predicted survival function S at every hour, starting at time 0.

```
f <- list(name = c('C','E','S'),
          time = seq(0,200,by=2))
```

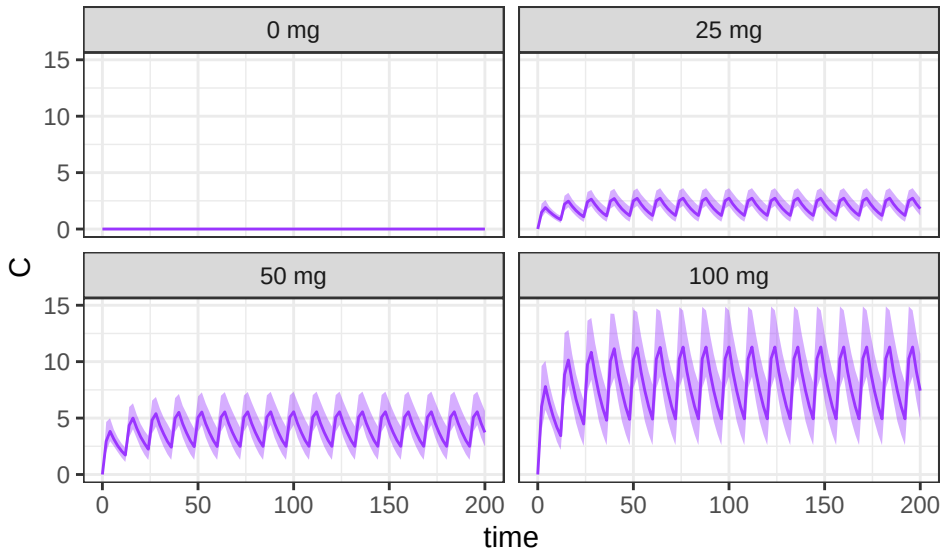
We can now run `simulx` with these input arguments:

```
res <- simulx(model     = "model/cts2II.txt",
              parameter = pop.param,
              group     = list(g1,g2,g3,g4),
              output    = f)
```

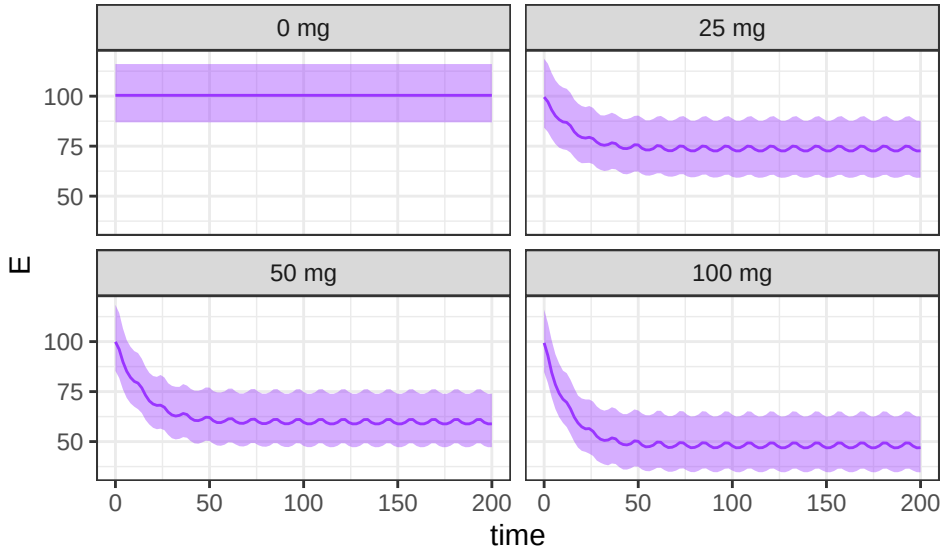
The list `res` contains the results of all the simulations.

We will use the function `prctilemlx` to compute and display several prediction intervals on a same plot using different shades of colour.

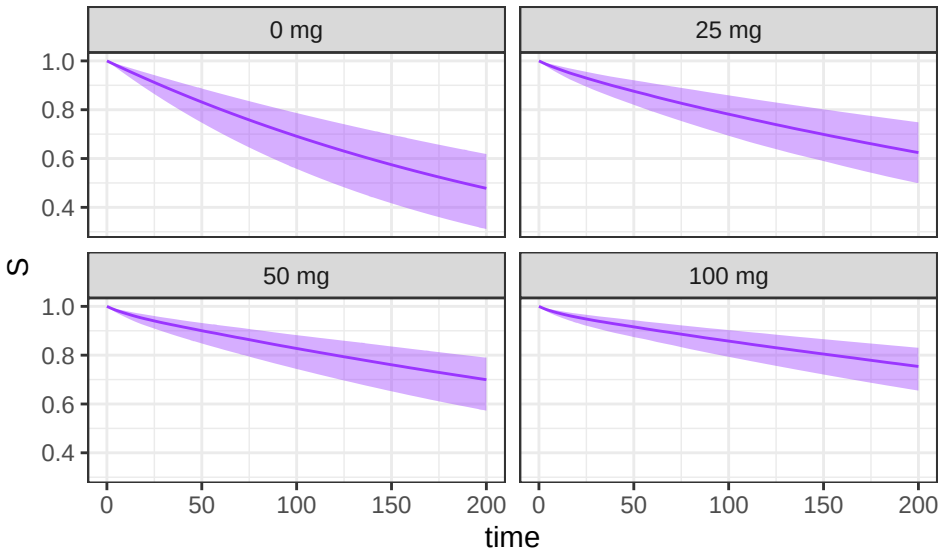
```
labels <- c("0 mg", "25 mg", "50 mg", "100 mg")
prctilemlx(res$C, number=2, level=90, labels=labels) + theme(legend.position = "none")
```




```
prctilemlx(res$E, number=2, level=90, labels=labels) + theme(legend.position = "none")
```




```
prctilemlx(res$S, number=2, level=90, labels=labels) + theme(legend.position = "none")
```



The simulations in this demonstration gave us the distribution of the concentration, disease progression and survival. However, in reality, where only a finite number of patients are included, this can not be observed. Therefore, in the next step to have more realistic simulations, a small number of patients will be simulated a number of times to see how the outcome varies between each trial simulation. This approach can be used to estimated the power of a clinical trial via simulations with `simulx`.

Comparing the effect from different dose-regimen on survival in cancer patients: part III

Task

The aim of this third part is to assess the information we can derive from a clinical trial with a particular design. One of the main characteristics of a clinical trial is that we have between-patient variation and a finite number of patients. Indeed, even if a treatement as an effect on survival as we have demonstrated in part I and part II of this example, a clinical trial with a limited number of patients will not necessarily be able to demonstrate this effect if a bad design has been selected.

It is therefore critical to assess the power of the trial. Here we show how we can compute the power by simulation of clinical trials with different number of patients and how we can compare the empirical survivals and the probability to conclude, correctly, that the treatment is effective.

The PKPD model

We use the same PKPD model as in part II. To have a more realistic scenario an observational error is added to the predicted concentration. This is done with an additional block DEFINITION in section [LONGITUDINAL] where the probability distributions of the outcomes are described.

We use here a normal distribution for the continuous outcome (concentration) and a survival model for the event. Survival is defined by a single event so the parameter `maxEventNumber` has to be set to one. The study has a finite duration. This means that we can not make any conclusions about survival after the study finished at 200 hours. In effect, this is a right censoring of the data and the parameter `rightCensoringTime` is set accorindly to 200.

This new joint PKPD model is implemented in the text file `model/cts2III.txt`:

```
[LONGITUDINAL]
input={Tk0,V,Cl,Imax,E0,IC50,kout,alpha,beta,b}

EQUATION:
C = pkmodel(Tk0, V, Cl)
g = b*C

E_0 = E0
kin = E0*kout
ddt_E = kin*(1-Imax*C/(C+IC50)) - kout*E

h = (alpha/1000)*exp(beta*E)

DEFINITION:
y = {distribution=normal, prediction=C, sd=g}
e = {type=event, maxEventNumber=1, rightCensoringTime=200, hazard=h}

;----------------------------------------
[INDIVIDUAL]
input={Tk0_pop,V_pop,Cl_pop,Imax_pop,E0_pop,IC50_pop,kout_pop,alpha_pop,beta_pop,omega_Tk0,omega_V,omega

DEFINITION:
Tk0   = {distribution=lognormal,   prediction=Tk0_pop,  sd=omega_Tk0}
V     = {distribution=lognormal,   prediction=V_pop,    sd=omega_V}
Cl    = {distribution=lognormal,   prediction=Cl_pop,   sd=omega_Cl}
E0    = {distribution=lognormal,   prediction=E0_pop,   sd=omega_E0}
IC50  = {distribution=lognormal,   prediction=IC50_pop, sd=omega_IC50}
```

```
kout  = {distribution=lognormal,   prediction=kout_pop, sd=omega_kout}
Imax  = {distribution=logitnormal, prediction=Imax_pop, sd=omega_Imax}
alpha = {distribution=normal,      prediction=alpha_pop,sd=omega_alpha}
beta  = {distribution=normal,      prediction=beta_pop, sd=omega_beta}
```

Clinical trial simulation

We investigate different trial designs different number of patients and different doses:

- $N = 25$, 50, and 100 patients,
- repeated oral administrations every 12 hours with different amounts: 0mg, 25mg, 50mg, 100mg.

Each trial simulation will be different because of the between-patient variability and the observational error. To understand the variability between different trial simulations we simulate $M = 1000$ replicates of each trial design, i.e. a total of $3 \times 4 \times 1000 = 12\,000$ trials.

```
M  <- 1000
vN <- c(25, 50, 100)
adm.amount <- c(0, 25, 50, 100)
adm.time   <- seq(0, 200, by=12)
```

Here, survival at t=100h and t=200h is the endpoint of interest. We will therefore use the event e defined in the model with the summary statistics `surv.t` defined at times 100 and 200.

```
e <- list(name='e', time=0, surv.time=c(100,200))
```

We use the same PKPD population parameters for each simulation:

```
pop.param   <- c(
  Tk0_pop   = 3,    omega_Tk0   = 0.2,
  V_pop     = 10,   omega_V     = 0.2,
  Cl_pop    = 1,    omega_Cl    = 0.2,
  Imax_pop  = 0.8,  omega_Imax  = 0.5,
  E0_pop    = 100,  omega_E0    = 0.1,
  IC50_pop  = 4,    omega_IC50  = 0.1,
  kout_pop  = 0.1,  omega_kout  = 0.1,
  alpha_pop = 0.5,  omega_alpha = 0.1,
  beta_pop  = 0.02, omega_beta  = 0,
  b         = 0.1
)
```

We use `simulx` for simulating M replicates of the trial, for each number of patients N and each dose.

The survival rates at times 100h and 200h for each simulated trial are stored in an array `R`.

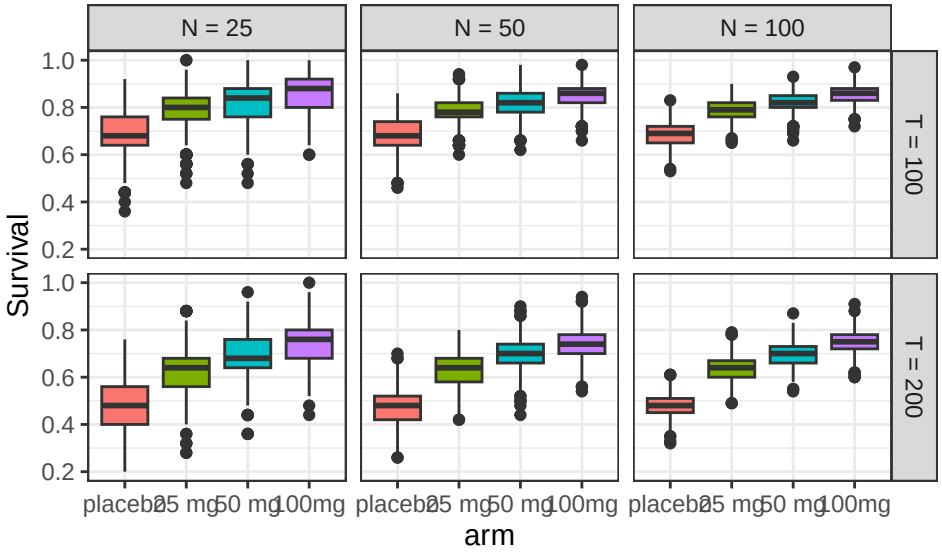
Important: In order to avoid any memory issue, we set `settings$out.trt = F` on purpose not to store the treatment given to each patient in each replicate.

```
labels.arm <- c("placebo", "25 mg", "50 mg", "100mg")
labels.N   <- c("N = 25", "N = 50", "N = 100")

R <- NULL
ptm <- proc.time()
for (k in seq(1,length(adm.amount))){
  adm <- list(time=adm.time, amount=adm.amount[k])
  for (l in seq(1,length(vN))){
  g <- list(size=vN[l], level='individual');
    res <- simulx(model     = "model/cts2III.txt",
                  parameter = pop.param,
```

```
                    treatment = adm,
                    output    = e,
                    group     = g,
                    nrep      = M,
                    settings  = list(out.trt = F))
    res$e$nbEv.mean <- NULL
    R <- rbind(R, cbind(res$e, N=labels.N[l], arm=labels.arm[k]))
  }
}
computational.time <- proc.time() - ptm
```

Once all the results are stored in array R, we can compare the survivals at time $t = 100$ and $t = 200$ using boxplots:

```
library(reshape2)
rm <- melt(R,  id = c('N','arm','rep'), value.name="Survival", variable.name="time")
rm$time <- factor(rm$time, labels = c("T = 100", "T = 200"))

print(ggplot(rm, aes(arm,Survival)) + geom_boxplot(aes(fill = arm)) +
        facet_grid(time ~N) + theme(legend.position="none"))
```



Imagine now that we conclude that the treatment is efficient if the difference in observed survival between the control group and the test group is greater than 0.1 at time 200.

The results of the simulation can now be used for estimating the power of the study, i.e. the probability to conclude, correctly, that the treatment is efficient, according to the dose and the size of the trial (we expect that this probability increases with both the dose and the size if the treatment is indeed efficient "in theory'').

```
S <- 0.1

R.p <- R[R$arm=="placebo","S200.mean"]
R.t <- R[R$arm!="placebo",]
R.t$dS <- (R.t$S200.mean-rep(R.p,3) > S)
R.t$S200.mean <- NULL
print(acast(R.t,arm~N,mean, value.var="dS"))
```

```
##       N = 25 N = 50 N = 100
## 25 mg  0.663  0.728   0.763
## 50 mg  0.795  0.890   0.949
## 100mg  0.911  0.965   0.990
```

Let us display the total computational time for simulating 12 000 trials on a labtop:

```
print(computational.time)
```

```
##    user  system elapsed
## 1283.34    8.23  340.22
```

Integrating simulx in a workflow a PK example

Introduction

`simulx` can be easily integrated in a workflow:

1. modeling of the **theophylline PK data** is performed first with `Monolix`,
2. simulation of PK data is then performed with `simulx`, using the parameters estimated by `Monolix`.

In this example we show how to simulate a model where the parameters have been estimated with `Monolix`. The Modelling and Simulation (M&S) workflow allows us to directly simulate the model just by providing the Monolix project as an input to `simulx`. Without any other inputs `simulx` will simulate the exact study design underlying the input data for parameter estimation. `simulx` gives us the flexibility to change any of the study design variables. We will demonstrate this by first simulating the original study design and then by changing observations, dosing regimens, number of patients and the patient covariates. Finally, we will show how to simulate a dose-concentration curve and how to do Monte Carlo simulations to assess how the number of patients affects the SE error of the concentration levels.

Examples

Define the project to be used for the simulations (relative paths)

```
project.file <- 'monolixRuns/theophylline_project.mlxtran'
```

1. Simulate a trial using the original design

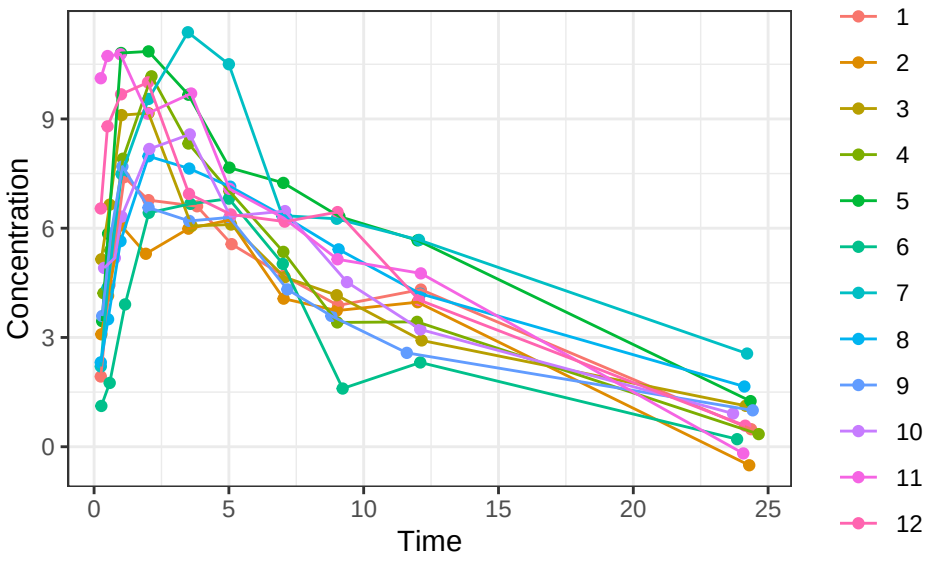
Simulate model with individual parameters sampled from the estimated population distribution

- Covariate WEIGHT is taken from data set
- Dosage regimen and observation times are taken from data set

```
sim.res1  <- simulx(project = project.file)
```

plot the results ‘’as usual”

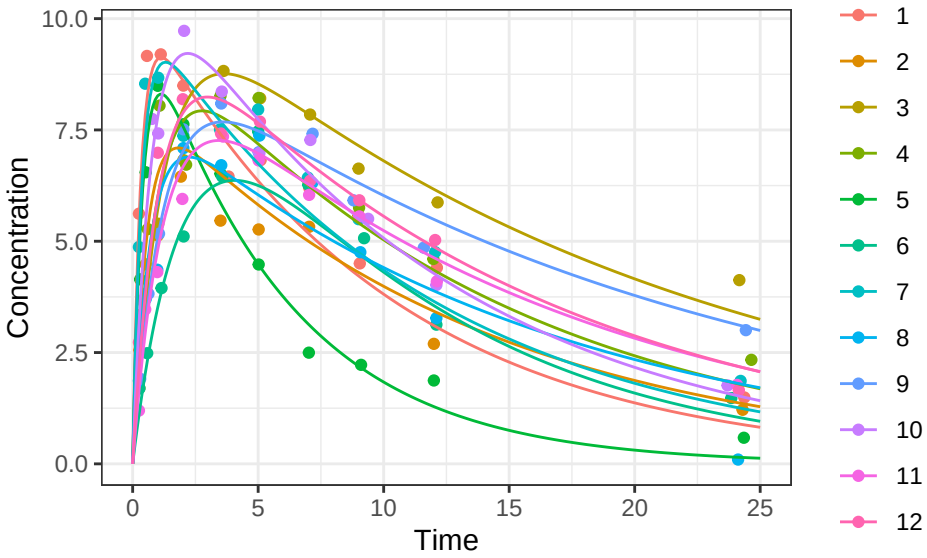
```
print(ggplot(data=sim.res1$y1) +
        geom_point(aes(x=time, y=y1, colour=id)) +
        geom_line(aes(x=time, y=y1, colour=id)) +
        scale_x_continuous("Time") + scale_y_continuous("Concentration"))
```



2. Remove the residual error

- Covariate WEIGHT is taken from data set
- Dosage regimen and observation times are taken from data set
- and switch off the residual error by setting b = 0

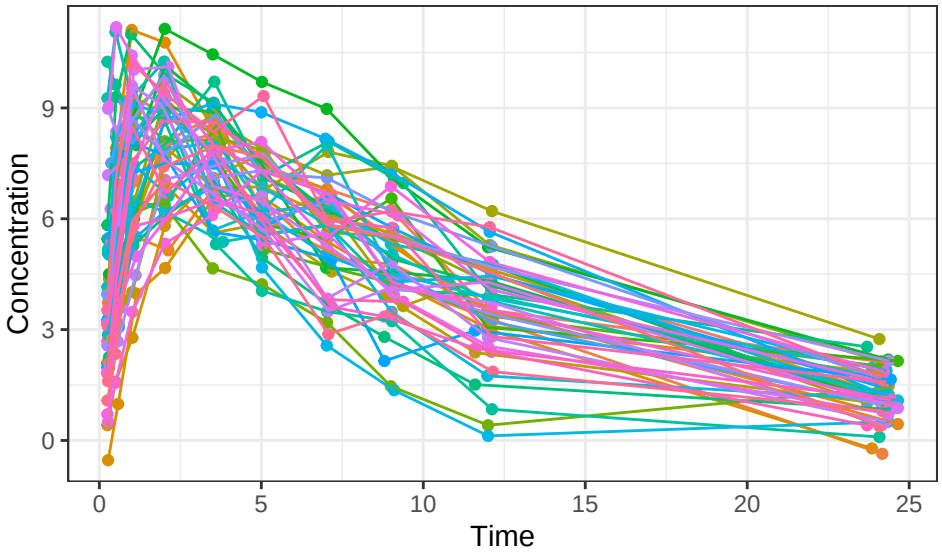
```
sim.param <- c(b=0)
out  <- list(name = 'Cc', time = seq(0, 25, by=0.1))
sim.res2  <- simulx(project    = project.file,
                    output     = out,
                    parameter = sim.param)

print(ggplot() +
        geom_point(data=sim.res2$y1, aes(x=time, y=y1, colour=id)) +
        geom_line(data=sim.res2$Cc, aes(x=time, y=Cc, colour=id)) +
        scale_x_continuous("Time") + scale_y_continuous("Concentration"))
```



3. Simulate a trial with N individuals

- The designs and the weights of these N patients are sampled from the original dataset

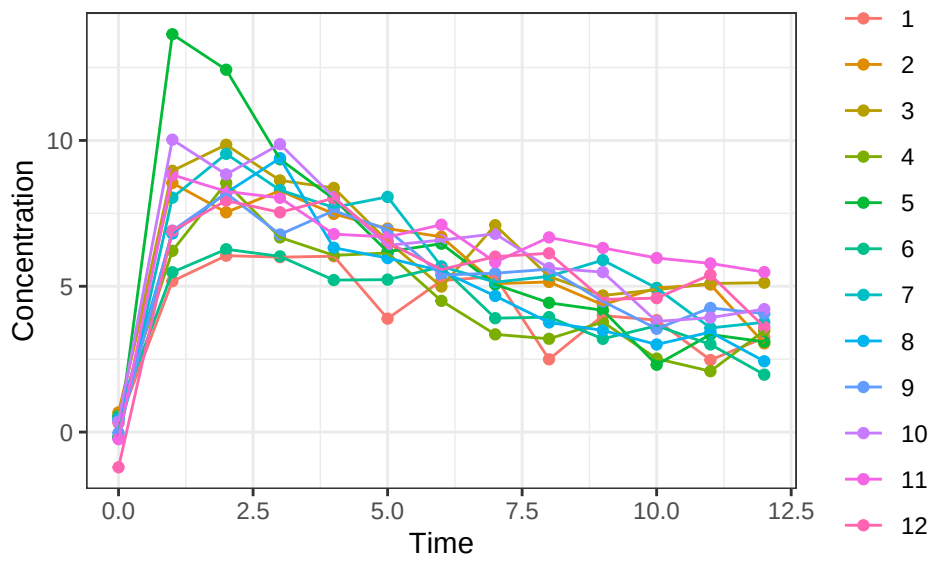
```
N <- 50
sim.res3  <- simulx(project = project.file,
                    group   = list(size = N))

print(ggplot(data=sim.res3$y1) +
        geom_point(aes(x=time, y=y1, colour=id)) +
        geom_line(aes(x=time, y=y1, colour=id)) +
        scale_x_continuous("Time") + scale_y_continuous("Concentration") +
        theme(legend.position="none"))
```



4. Define new observation times

- Covariate WEIGHT and dosage regimen are taken from data set

```
out  <- list(name = 'y1', time = (0:12))

sim.res4  <- simulx(project = project.file,
                    output = out)

print(ggplot(data=sim.res4$y1) +
        geom_point(aes(x=time, y=y1, colour=id)) +
        geom_line(aes(x=time, y=y1, colour=id)) +
        scale_x_continuous("Time") + scale_y_continuous("Concentration"))
```



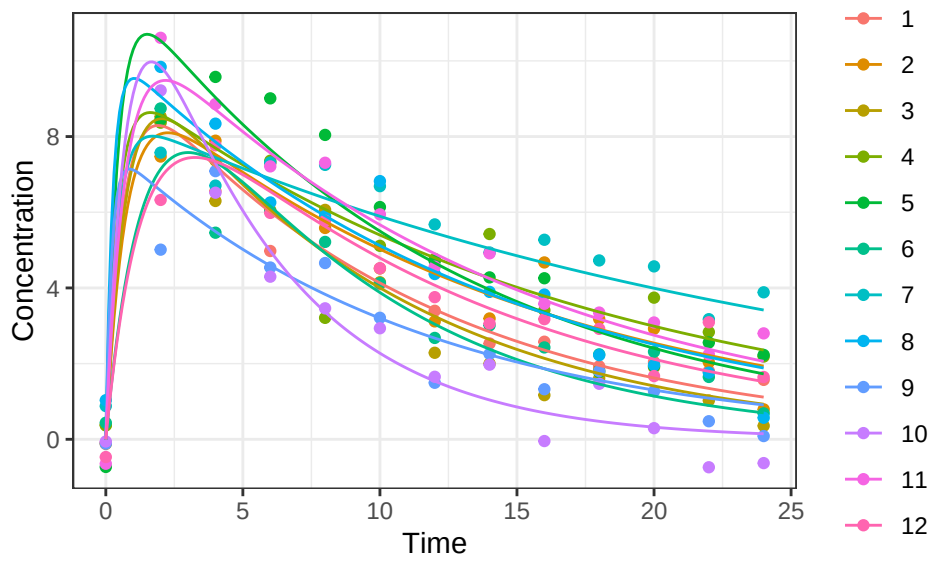
5. Output the predicted concentrations, the individual parameters and the covariates

- Covariate WEIGHT and dosage regimen are taken from data set

```
out1  <- list(name = 'y1', time = seq(0, 24, by=2))
out2  <- list(name = 'Cc', time = seq(0, 24, by=0.1))
out3  <- list(name = c('V', 'Cl', 'WEIGHT'))

sim.res5  <- simulx(project = project.file,
                    output  = list(out1, out2, out3))

print(sim.res5$parameter)
```

```
##    id        V       Cl WEIGHT
## 1   1 32.29205 3.021865   79.6
## 2   2 33.85839 2.307886   72.4
## 3   3 30.49988 3.183391   70.5
## 4   4 33.67618 1.991105   72.7
## 5   5 26.39646 2.182823   54.6
## 6   6 29.22527 3.581980   80.0
## 7   7 37.37728 1.455519   64.6
## 8   8 31.16473 2.225773   70.5
## 9   9 41.46337 3.729925   86.4
## 10 10 23.17127 4.537883   58.2
## 11 11 28.83027 2.084296   65.0
## 12 12 32.93698 2.694038   60.5
```

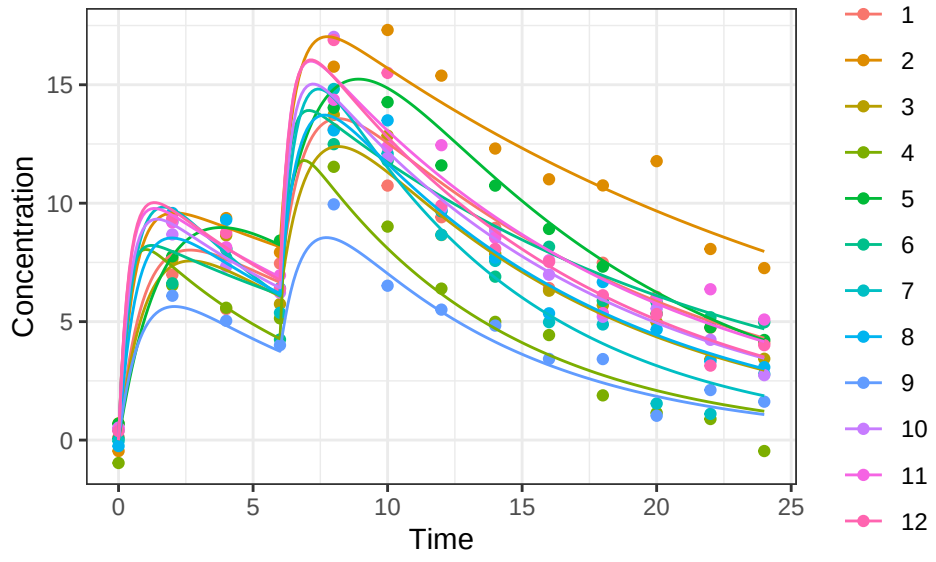
```
print(ggplot() +
        geom_point(data=sim.res5$y1,aes(x=time, y=y1, colour=id)) +
        geom_line(data=sim.res5$Cc,aes(x=time, y=Cc, colour=id)) +
        scale_x_continuous("Time") + scale_y_continuous("Concentration"))
```



6. Specify the administration schedule

- Covariate WEIGHT is taken from data set
- Define observation times for several given outputs

```
adm   <- list(time = c(0,6), amount = c(320, 320))

sim.res6  <- simulx(project    = project.file,
                    treatment = adm,
                    output     = list(out1, out2))

print(ggplot() +
        geom_point(data=sim.res6$y1,aes(x=time, y=y1, colour=id)) +
        geom_line(data=sim.res6$Cc,aes(x=time, y=Cc, colour=id)) +
        scale_x_continuous("Time") + scale_y_continuous("Concentration"))
```

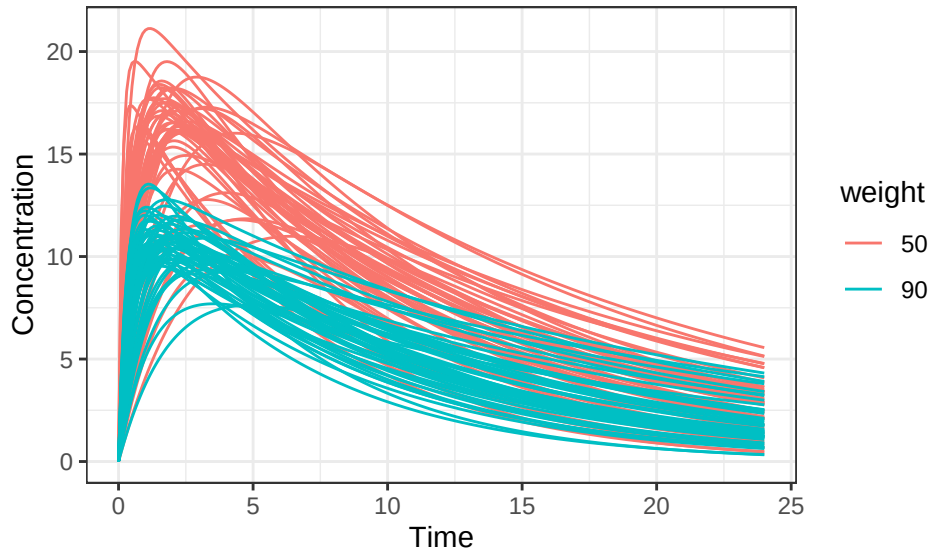


7. Specify own WEIGHT covariates for N patients

- define observation times for several given outputs
- and specify the administration schedule
- and specify own WEIGHT covariates

- and increase patient number from the original 12 to 100.

```
N       <- 100
weight <- list( name     = 'WEIGHT',
                colNames = c('id', 'WEIGHT'),
                value    = cbind(c(1:N),c(rep(50, N/2), rep(90, N/2))))

adm   <- list(time = 0, amount = 500)

sim.res7 <- simulx(project = project.file,
                   output = list(out1, out2),
                   treatment = adm,
                   parameter = weight)

sim.res7$Cc$weight <- 50
sim.res7$Cc$weight[as.numeric(sim.res7$Cc$id)>N/2] <-90
sim.res7$Cc$weight <- as.factor(sim.res7$Cc$weight)

print(ggplot() +
        geom_line(data=sim.res7$Cc,aes(x=time, y=Cc, group=id, colour=weight)) +
        scale_x_continuous("Time") + scale_y_continuous("Concentration"))
```

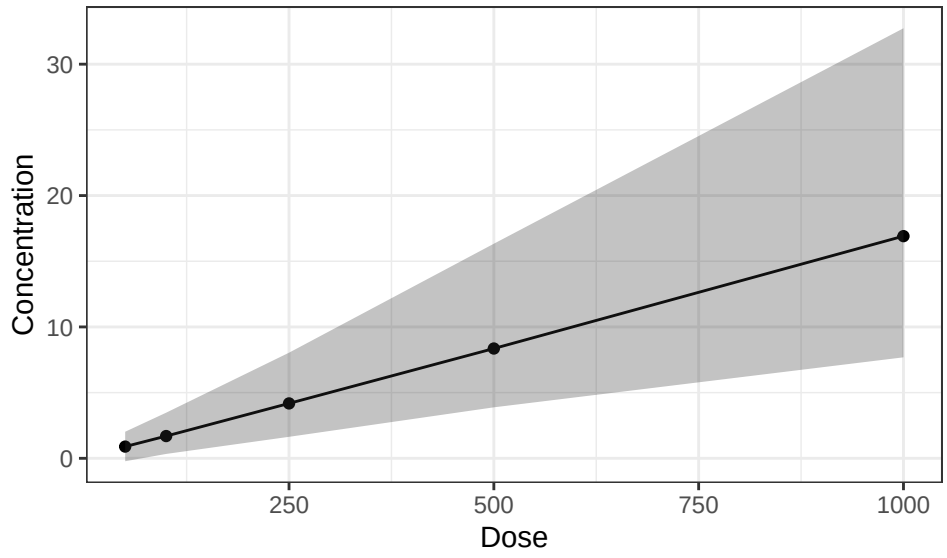


8. Estimate the dose-concentration relationship

- select an observation time for the concentration
- specify own WEIGHT covariates

- increase patient number from the original 12 to 100.
- specify several administration schedules
- use the same individuals for each dose

```
library(plyr, warn.conflicts = FALSE)

#-- Define the number of patients to simulate
N <- 100

#-- Generate individual weights by sampling a lognormal distribution
sim.weight <- data.frame(id=1:N, WEIGHT=rlnorm(N, log(70), 0.2))

outy  <- list(name = 'y1', time = 120)
Dose.amount <- c(50, 100, 250, 500, 1000)
Dose.times  <- seq(from = 0, to = 120, by = 12)

#-- Use the same patients for each of the dose levels
s  <- list(seed = 123456)

#-- run simulx with each dose level
sim.data <- NULL
for(n.c in 1:length(Dose.amount)) {
  adm   <- list(time = Dose.times, amount = Dose.amount[n.c])

    tmp <- simulx(project   = project.file,
                  output    = outy,
                  treatment = adm,
                  parameter = sim.weight,
                  settings  = s)

    tmp2            <- tmp$y1
    tmp2['Dose']    <- Dose.amount[n.c]
    sim.data <- rbind(sim.data, tmp2)
}

#-- Compute statistics
sim.data.stat <- ddply(sim.data, .(time, Dose), summarize,
    median = median(y1),
    p05  = quantile(y1, 0.05),
    p95  = quantile(y1, 0.95),
    Dose = Dose[1])

print(ggplot(data=sim.data.stat) +
  geom_line(aes(x=Dose, y=median)) +
  geom_point(aes(x=Dose, y=median)) +
  geom_ribbon(aes(x=Dose, ymin=p05, ymax=p95), alpha = 0.3) +
  scale_x_continuous("Dose") + scale_y_continuous("Concentration"))
```



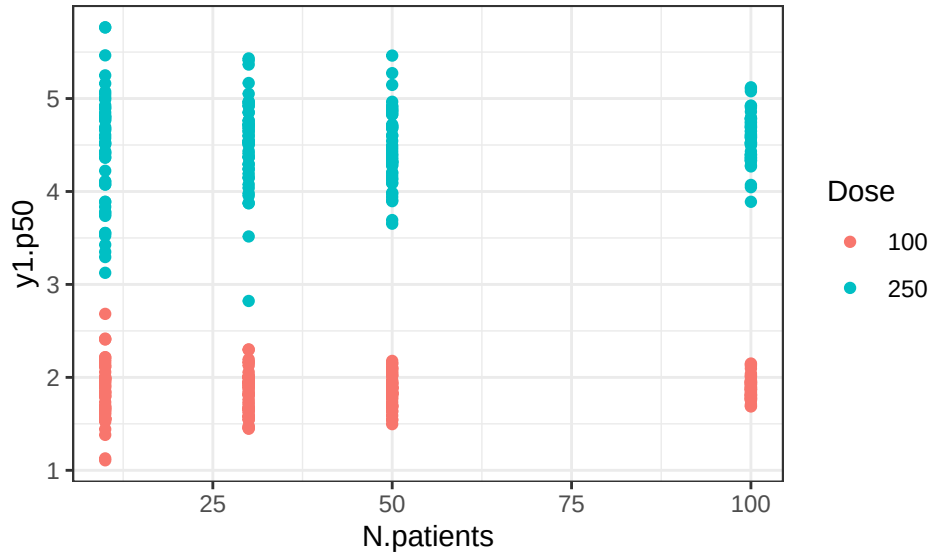
9. Evaluate how the sample size and the dose affect the distribution of the concentration

For each sample size and each dose, we run multiple trial simulations and calculate several quantiles; Weights are sampled from the original individual weights.

```
#-- Define the output as summary statistics of the simulated concentration at t=120
out.y <- list(name='y1', time=120, FUN = "quantile", probs = c(0.05, 0.50, 0.95))

Dose.amount <- c(100, 250)
Dose.times  <- seq(from = 0, to = 120, by = 12)

#-- Define the number of trial simulations
N.trial <- 50

#-- Define the number of patients to simulate
vN <- c(10, 30, 50, 100)

sim.data <- NULL
for(n.c in 1:length(Dose.amount)){
  adm   <- list(time = Dose.times, amount = Dose.amount[n.c])
  for(N in vN){

    tmp1 <- simulx(project   = project.file,
                   nrep = N.trial,
                   output    = out.y,
                   treatment = adm,
                   group = list(size=N))

    tmp2 <- tmp1$y1
    tmp2['Dose'] <- factor(Dose.amount[n.c])
    tmp2['N.patients']<- N
    sim.data <- rbind(sim.data, tmp2)
  }
}
head(sim.data)
```

```
##   rep time     y1.p5   y1.p50   y1.p95 Dose N.patients
## 1   1  120 0.4734836 1.889001 3.236321  100         10
## 2   2  120 1.0791291 1.552569 2.832704  100         10
## 3   3  120 0.8083024 2.171027 3.140980  100         10
## 4   4  120 1.2069917 1.562615 3.863982  100         10
## 5   5  120 0.5426089 1.918455 4.863036  100         10
## 6   6  120 1.0437746 2.007906 2.653437  100         10
```

```
## Distribution of the median
print(ggplot(data=sim.data) + geom_point(aes(x=N.patients, y=y1.p50, color=Dose)))
```

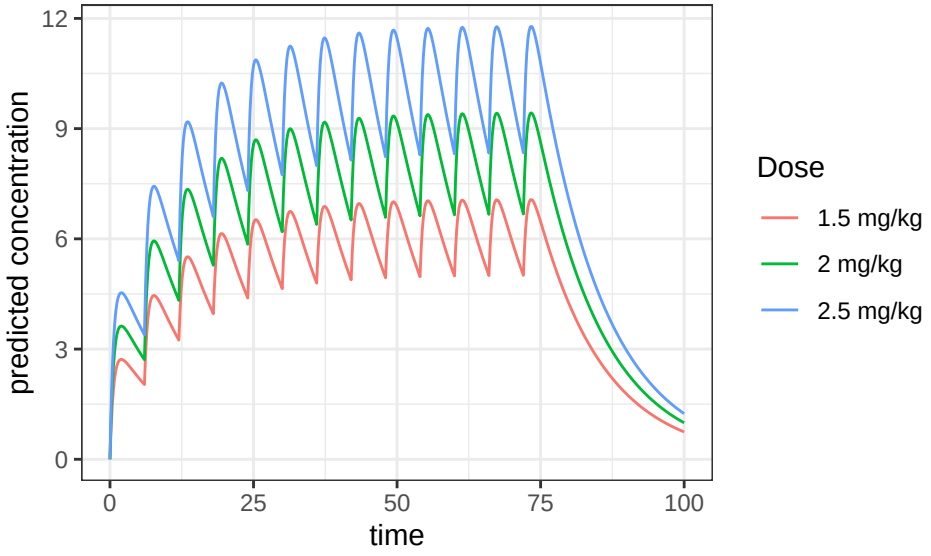


10. Compare predicted concentrations for different doses

- Compute the predicted concentration Cc when 1.5mg/kg, 2mg/kg or 2.5mg/kg are given every 6 hours to a typical patient with 70kg and no random effects.

```
WEIGHT = 70
sim.param <- c(omega_V=0, omega_ka=0, omega_Cl=0, WEIGHT=WEIGHT)  # define a typical patient

out.Cc  <- list(name = 'Cc', time = seq(0,100,by=0.2))

Dose.times  <- seq(from = 0, to = 72, by = 6)
nb.times <- length(Dose.times)

Dose.perkg <- c(1.5, 2, 2.5)
nb.group <- length(Dose.perkg)

g <- list()
for (k in (1:nb.group)) {
  trt.k <- list(time = Dose.times, amount = round(Dose.perkg[k]*WEIGHT))
  g[[k]] <- list(treatment = trt.k, size=1)
}

res <- simulx(project   = project.file,
              group     = g,
              parameter = sim.param,
```

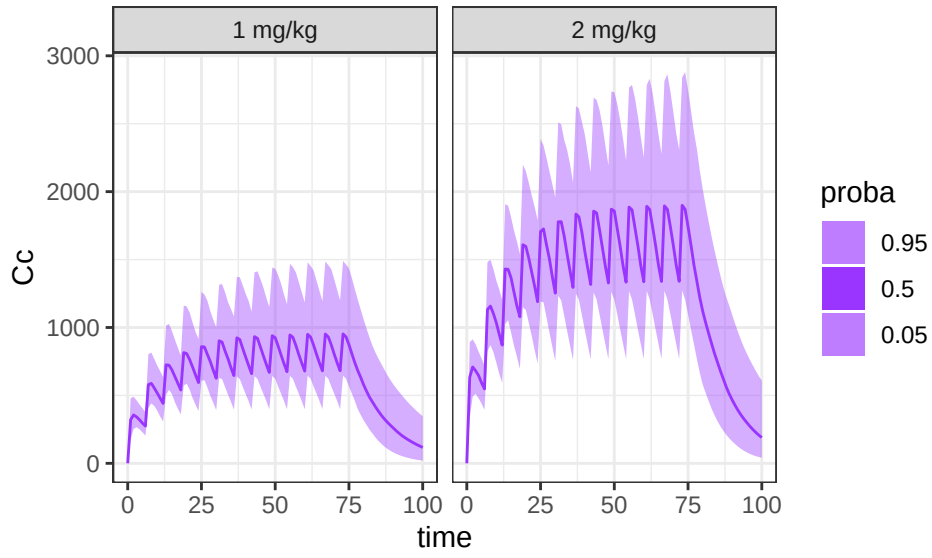
```
              output    = out.Cc)
pl <- ggplot(data=res$Cc, ) + geom_line(aes(x=time, y=Cc, colour=group)) +
  scale_y_continuous("predicted concentration") +
  scale_colour_discrete(name  ="Dose", labels=c("1.5 mg/kg", "2 mg/kg", "2.5 mg/kg"))
print(pl)
```



11. Compare prediction intervals of the concentration for different doses

- Compute by Monte Carlo 90% prediction intervals for the predicted concentration Cc when 1mg/kg or 2mg/kg are given every 6 hours.
- Individual weights are simulated.

```
N <- 200
sim.weight <- data.frame(id=1:N, WEIGHT=rlnorm(N, log(70), 0.2))

out.Cc  <- list(name = 'Cc', time = 0:100)
Dose.times  <- seq(from = 0, to = 72, by = 6)
nb.times <- length(Dose.times)
Dose.perkg <- c(1, 2)
nb.group <- length(Dose.perkg)
g <- list()
for (k in (1:nb.group)) {
  trt.k <- data.frame(id     = rep(1:N,each=nb.times),
                      time   = rep(Dose.times,N),
                      amount = rep(round(Dose.perkg[k]*sim.weight$WEIGHT),each=nb.times))
  g[[k]] <- list(treatment = trt.k, size=N)
}

res <- simulx(project   = project.file,
              group     = g,
              parameter = sim.weight,
              output    = out.Cc)
pl <- prctilemlx(res$Cc, number = 2, level = 90, labels = c("1 mg/kg", "2 mg/kg"))
print(pl)
```


1 mg/kg
2 mg/kg
3000
2000
1000
0
Cc
0
25
50
75
100
time
proba
0.95
0.5
0.05

Integrating simulx in a workflow a PKPD example

Introduction

In this example,

1. modeling of the **warfarin PKPD data** is performed first with `Monolix`,
2. simulation of PKPD data is then performed with `simulx`, using the parameters estimated by `Monolix`.

Define the project to be used for the simulations (relative paths)

```
project.file <- 'monolixRuns/warfarin_project.mlxtran'
```

Examples

Example 1

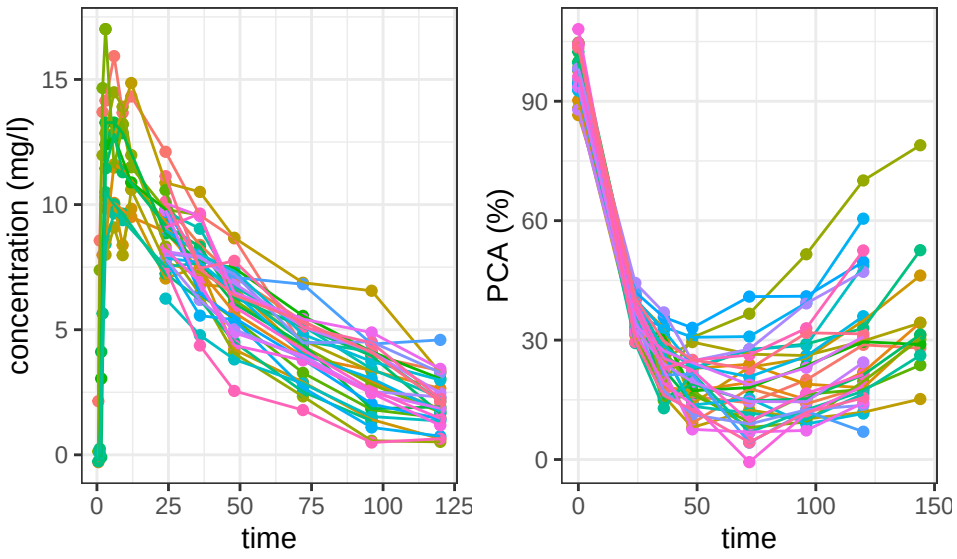
Simulate model with individual parameters sampled from the estimated population distribution

- Covariate WEIGHT is taken from data set
- Dosage regimen and observation times are taken from data set

```
res <- simulx(project=project.file)
```

plot the results ‘’as usual”

```
library(gridExtra)

plot1 <- ggplot() + geom_point(data=res$y1, aes(x=time, y=y1, colour=id)) +
  geom_line( data=res$y1, aes(x=time, y=y1, colour=id)) +
  theme(legend.position="none") + ylab("concentration (mg/l)")
plot2 <- ggplot() + geom_point(data=res$y2, aes(x=time, y=y2, colour=id)) +
  geom_line( data=res$y2, aes(x=time, y=y2, colour=id)) +
  theme(legend.position="none") + ylab("PCA (%)")
grid.arrange(plot1,plot2,ncol=2)
```

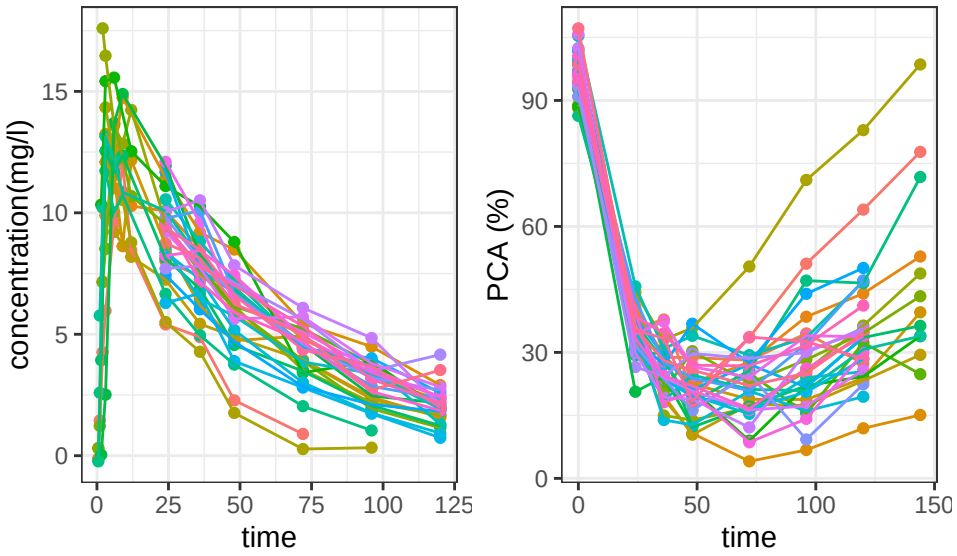


Example 2

- Estimated individual parameters (conditional modes) are used the simulations

- Dosage regimen and observation times are taken from data set

```
res <- simulx( project   = project.file,
               parameter = 'mode')

plot1 <- ggplot() +
  geom_point(data=res$y1, aes(x=time, y=y1, colour=id)) +
  geom_line( data=res$y1, aes(x=time, y=y1, colour=id)) +
  theme(legend.position="none") + ylab("concentration(mg/l)")
plot2 <- ggplot() +
  geom_point(data=res$y2, aes(x=time, y=y2, colour=id)) +
  geom_line( data=res$y2, aes(x=time, y=y2, colour=id)) +
  theme(legend.position="none") + ylab("PCA (%)")
grid.arrange(plot1,plot2,ncol=2)
```

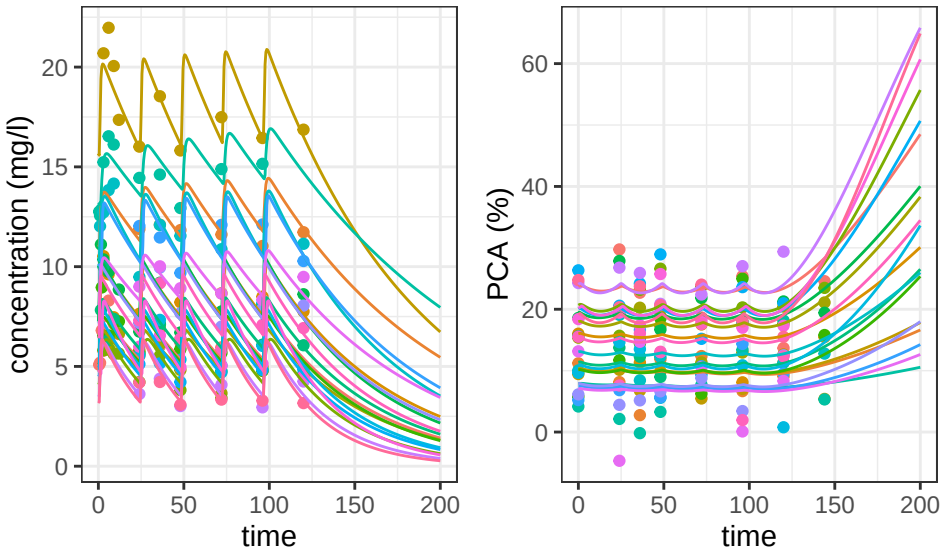


Example 3

- simulate a trial with N=20 individuals: the designs and the weights of these N patients are sampled from the original dataset with replacement
- and define predicted PK and PD as outputs
- and specify the administration schedule

```
N   <- 20
adm <- list( amount = 25, time = seq(-240, 96, by=24 ));
out <- list( name = c('Cc','E'), time = seq(0, 200, by=0.5));
res <- simulx( project   = project.file,
               output    = out,
               treatment = adm,
               group     = list(size=N))

plot1 <- ggplot() + geom_point(data=res$y1, aes(x=time, y=y1, colour=id)) +
  geom_line( data=res$Cc, aes(x=time, y=Cc, colour=id)) +
  theme(legend.position="none") + ylab("concentration (mg/l)")
plot2 <- ggplot() + geom_point(data=res$y2, aes(x=time, y=y2, colour=id)) +
  geom_line( data=res$E, aes(x=time, y=E, colour=id)) +
  theme(legend.position="none") + ylab("PCA (%)")
```

```
grid.arrange(plot1,plot2,ncol=2)
```



Integrating simulx in a workflow An example with continuous PK and categorical PD data

In this example,

1. modeling of the **modified warfarin PKPD data** (the original continuous PD data has transformed into categorical data with levels {0, 1, 2}) is performed first with `Monolix`,
2. simulation of joint continuous and categorical data is then performed with `simulx`, using the parameters estimated by `Monolix`.

Define the project to be used for the simulations (relative paths)

```
project.file <- 'monolixRuns/pkcat_project.mlxtran'
```

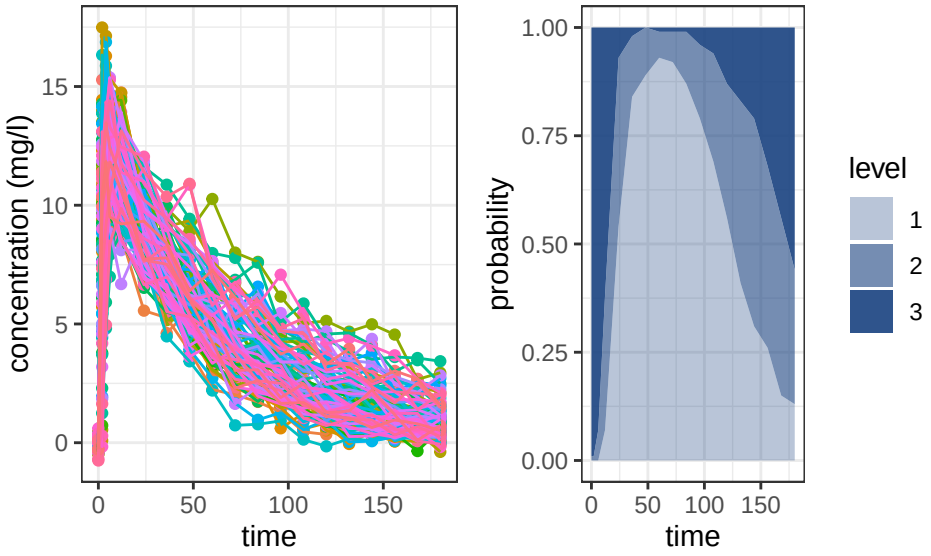
Simulate a trial with N=20 individuals:

- the designs and the weights of these N patients are sampled from the original dataset with replacement
- individual parameters are sampled from the estimated population distribution.

```
N=100
out  <- list(name = c('conc','level'), time = c(0,2,4,6,seq(12, 180, by=12)))
p <- c(a_1=0, b_1=0)

res <- simulx(project=project.file,
              group     = list(size = N),
              output    = out)
```

Plot the simulated PK data and the distribution of the simulated categorical PD data

```
library(gridExtra)

plot1 <- ggplot(data=res$conc) + geom_point(aes(x=time, y=conc, colour=id)) +
  geom_line(aes(x=time, y=conc, colour=id)) +
  theme(legend.position="none") + ylab("concentration (mg/l)")

plot2 <- catplotmlx(res$level)

grid.arrange(plot1, plot2, ncol=2)
```



Integrating simulx in a workflow An example with continuous PK and repeated events

In this example,

1. modeling of **simulated joint data** (continuous data and repeated events) is performed first with `Monolix`,
2. simulation of joint continuous data and repeated events is then performed with `simulx`, using the parameters estimated by `Monolix`.

Define the project to be used for the simulations (relative paths)

```
project.file <- 'monolixRuns/pkrtte_project.mlxtran'
```

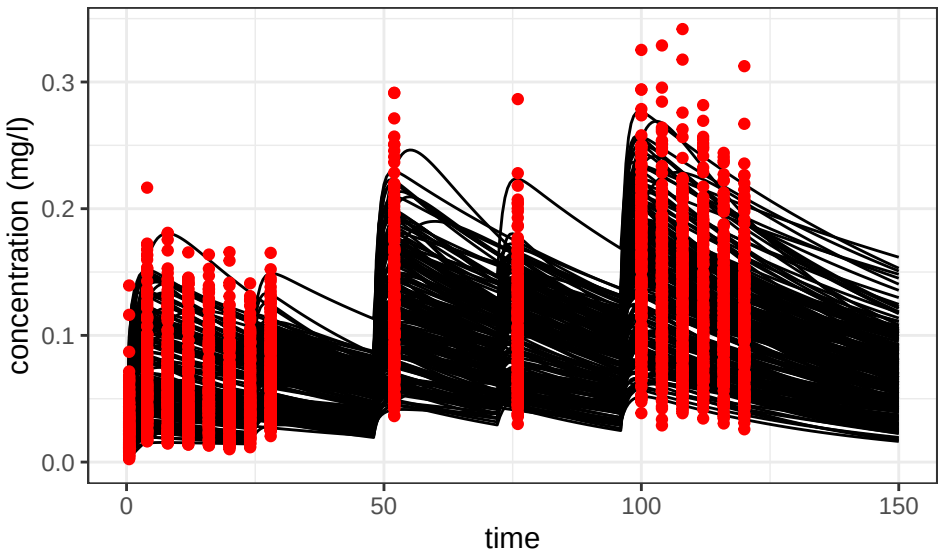
Simulate a trial with N=200 individuals:

- the designs of these N patients are sampled from the original dataset with replacement
- individual parameters are sampled from the estimated population distribution
- the predicted concentration is defined as output.

```
N=200
out  <- list(name = c('Cc'), time = seq(0,150,by=0.5))
res <- simulx(project=project.file,
              group     = list(size = N),
              output    = out)
```

Plot the simuated PK data and the predicted concentrations

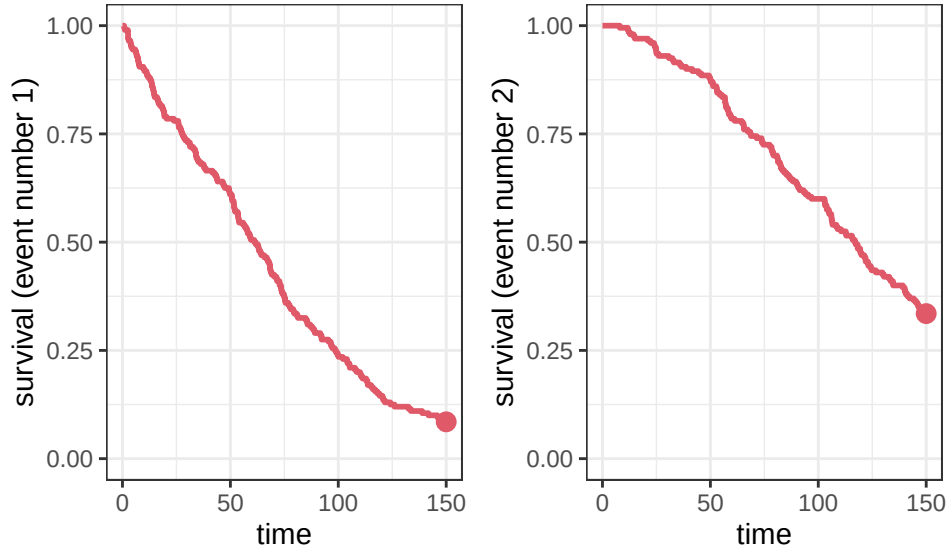
```
plot1 <- ggplot() +
  geom_line(data=res$Cc, aes(x=time, y=Cc, group=id), colour="black") +
  geom_point(data=res$Concentration, aes(x=time, y=Concentration), colour="red") +
  theme(legend.position="none") + ylab("concentration (mg/l)")
print(plot1)
```



Plot the survival functions for the first and second events

```
library("gridExtra")
plot2a <- kmplotmlx(res$Hemorrhaging) + ylim(c(0,1))
```

```
plot2b <- kmplotmlx(res$Hemorrhaging, index=2) + ylim(c(0,1))
grid.arrange(plot2a,plot2b,ncol=2)
```



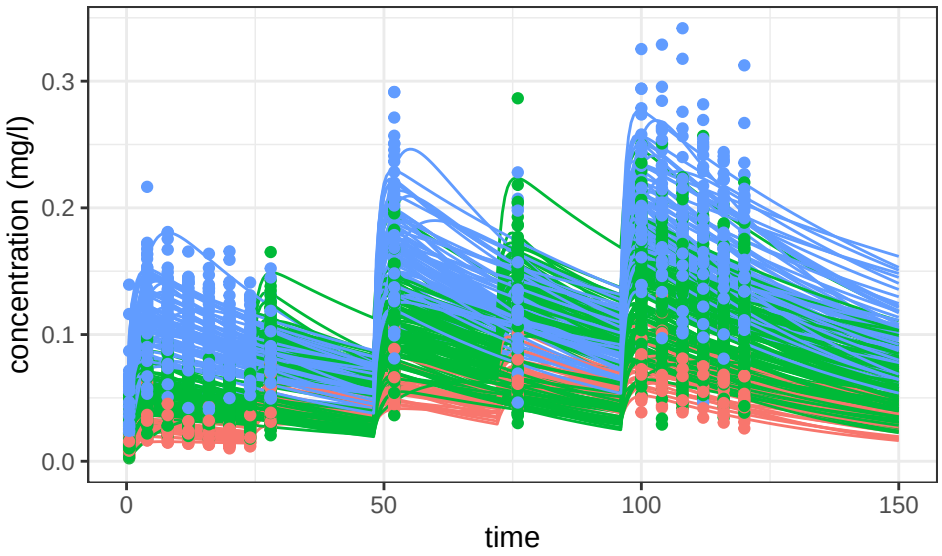
Three different doses are administrated in this clinical trial: 0.25g, 0.5 and 1g.

We can therefore add the amount to the simulated data

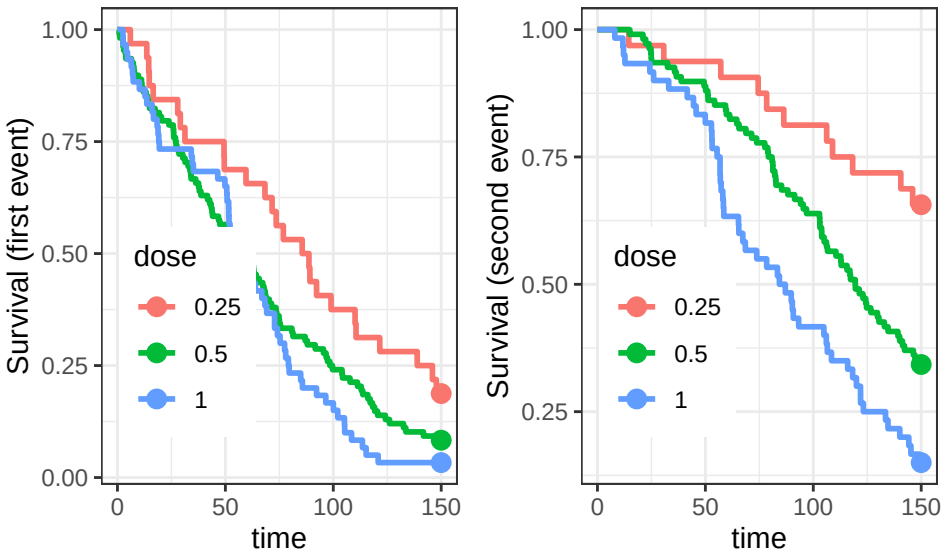
```
trt0 <- res$treatment[res$treatment$time==0,c(1,3)]
rh <- merge(res$Hemorrhaging,trt0)
ry <- merge(res$Concentration,trt0)
rc <- merge(res$Cc,trt0)
```

and distinguish these three treatment groups in the plots:

```
plot3 <- ggplot() +
  geom_line(data=rc, aes(x=time, y=Cc, group=id, colour=factor(amount))) +
  geom_point(data=ry, aes(x=time, y=Concentration, colour=factor(amount))) +
  theme(legend.position="none") + ylab("concentration (mg/l)")
print(plot3)
```




```
plot4a <- kmplotmlx(rh, group="amount", facet=FALSE)  + ylab("Survival (first event)") +
  scale_colour_discrete(name  ="dose")
plot4b <- kmplotmlx(rh, index=2, group="amount", facet=FALSE)  + ylab("Survival (second event)") +
  scale_colour_discrete(name  ="dose")
grid.arrange(plot4a,plot4b,ncol=2)
```



Integrating simulx in a workflow An example with censored data

A PK example with BLQ data

In this first example, we use the theophylline project used previously.

Let us start simulting new PK data without any censoring process.

```
project1 <- 'monolixRuns/theophylline_project.mlxtran'

res0 <- simulx(project  = project1,
               settings = list(seed=12321) )
head(res0$y1, n=9)
```

```
##   id  time       y1
## 1  1  0.25 1.767205
## 2  1  0.57 3.563156
## 3  1  1.12 4.682249
## 4  1  2.02 6.036410
## 5  1  3.82 7.262963
## 6  1  5.10 6.611404
## 7  1  7.03 4.472900
## 8  1  9.05 5.079954
## 9  1 12.12 3.228050
```

We can easily introduce a lower limit of quantification LLOQ=2 by redifing the output `y1`. Since the sampling times are not defined, the original ones are used.

```
out.y <- list(name="y1", lloq=2)
res1 <- simulx(project  = project1,
               output   = out.y,
               settings = list(seed=12321) )
head(res1$y1, n=10)
```

```
##    id  time       y1 cens
## 1   1  0.25 2.000000    1
## 2   1  0.57 3.563156    0
## 3   1  1.12 4.682249    0
## 4   1  2.02 6.036410    0
## 5   1  3.82 7.262963    0
## 6   1  5.10 6.611404    0
## 7   1  7.03 4.472900    0
## 8   1  9.05 5.079954    0
## 9   1 12.12 3.228050    0
## 10  1 24.37 2.000000    1
```

New sampling times can also be defined together with LLOQ

```
out.y <- list(name="y1", time=seq(0,24,by=2), lloq=2)
res2 <- simulx(project  = project1,
               output   = out.y,
               settings = list(seed=12321) )
head(res2$y1, n=10)
```

```
##    id time       y1 cens
## 1   1    0 2.000000    1
## 2   1    2 6.649227    0
## 3   1    4 6.365703    0
```

```
## 4   1    6 5.749798    0
## 5   1    8 5.786707    0
## 6   1   10 4.827677    0
## 7   1   12 2.939633    0
## 8   1   14 3.650868    0
## 9   1   16 2.341326    0
## 10  1   18 2.268264    0
```

The BLQ data are known to take positive values. Then, an additional column LIMIT=0 can be added to the output,

```
out.y$limit <- 0
res3 <- simulx(project  = project1,
               output   = out.y,
               settings = list(seed=12321) )
head(res3$y1, n=10)
```

```
##    id time       y1 cens limit
## 1   1    0 2.000000    1     0
## 2   1    2 6.649227    0     0
## 3   1    4 6.365703    0     0
## 4   1    6 5.749798    0     0
## 5   1    8 5.786707    0     0
## 6   1   10 4.827677    0     0
## 7   1   12 2.939633    0     0
## 8   1   14 3.650868    0     0
## 9   1   16 2.341326    0     0
## 10  1   18 2.268264    0     0
```

A PKPD example with BLQ and ALQ data

We will use the warfarin project for this second example.

We define a lower limit of quantification LLOQ=8 for the PK and an upper limit ULOQ=40 for the PD. We also take into account the fact that the PK data take positive value by defining LIMIT=0.

CENS takes the value 1 for BLQ data and -1 for data above the limit of quantification (ALQ data). Otherwise, CENS=0.

```
project2 <- 'monolixRuns/warfarin_project.mlxtran'
y1 <- list(name="y1", lloq=8, limit=0)
y2 <- list(name="y2", uloq=40)

res4 <- simulx(project  = project2,
               output   = list(y1, y2),
               settings = list(seed=123) )
head(res4$y1, n=11)
```

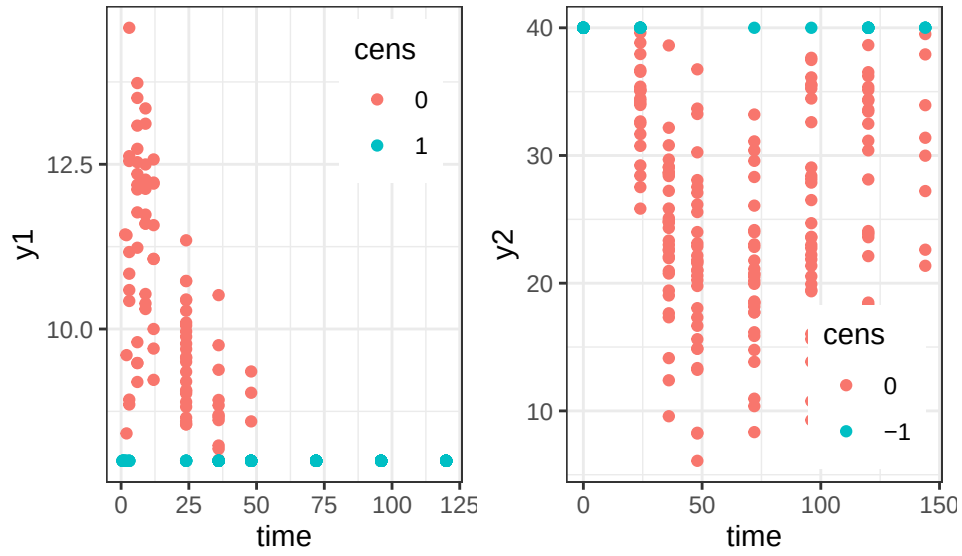
```
##    id time        y1 cens limit
## 1   1  0.5  8.000000    1     0
## 2   1  1.0  8.000000    1     0
## 3   1  2.0  8.000000    1     0
## 4   1  3.0  8.000000    1     0
## 5   1  6.0 13.086700    0     0
## 6   1  9.0 11.736617    0     0
## 7   1 12.0 12.206153    0     0
## 8   1 24.0  8.594241    0     0
```

```
## 9   1 36.0  8.000000    1     0
## 10  1 48.0  8.000000    1     0
## 11  1 72.0  8.000000    1     0
```

```
head(res4$y2, n=7)
```

```
##   id time       y2 cens
## 1  1   24 40.00000   -1
## 2  1   36 24.79613    0
## 3  1   48 36.74814    0
## 4  1   72 40.00000   -1
## 5  1   96 40.00000   -1
## 6  1  120 40.00000   -1
## 7  1  144 40.00000   -1
```

```
library(gridExtra)
pl1 <- ggplot() +
  geom_point(data=res4$y1, aes(x=time, y=y1, colour=cens, group=id)) +
  theme(legend.position=c(.8, .8))
pl2 <- ggplot() +
  geom_point(data=res4$y2, aes(x=time, y=y2, colour=cens, group=id)) +
  theme(legend.position=c(.8, .2))
grid.arrange(pl1, pl2, ncol=2)
```



Integrating simulx in a workflow An example with inter occasion variability

In this first example, we use the Monolix project iov_project.mlxtran.

```
iov.project <- 'monolixRuns/iov_project.mlxtran'

d <- readDatamlx(project = iov.project)
print(names(d))
```

```
##  [1] "populationParameters" "treatment"            "y1"
##  [4] "occasion"             "covariate.iov"        "id"
##  [7] "N"                    "catNames.iov"         "covariate.iiv"
## [10] "catNames.iiv"
```

1, 2 or 3 occasions of length 12 hours are defined for each individual in this project:

```
head(d$occasion, 13)
```

```
##     id time occ
## 1    1    0   1
## 15   1   24   2
## 29   2    0   1
## 42   2   24   2
## 55   3    0   1
## 64   3   24   2
## 73   4    0   1
## 83   5    0   1
## 93   5   24   2
## 103  6    0   1
## 113  6   24   2
## 123  6   48   3
## 133  7    0   1
```

Covariates C1 and C3 do not change over time

```
head(d$covariate.iiv)
```

```
##   id C1 C3
## 1  1  5  A
## 2  2 10  A
## 3  3 15  A
## 4  4 20  A
## 5  5 25  A
## 6  6 30  A
```

while C2 and C4 change over time. Note that the column OCC which defines the occasions is also defined as a time varying covariate

```
head(d$covariate.iov,12)
```

```
##     id time C2 C4 OCC
## 1    1    0  7  A   1
## 15   1   24 12  B   2
## 29   2    0  9  A   1
## 42   2   24 14  B   2
## 55   3    0 11  A   1
## 64   3   24 16  B   2
```

```
## 73   4    0 13  A   1
## 83   5    0 15  A   1
## 93   5   24 20  B   2
## 103  6    0 17  A   1
## 113  6   24 22  B   2
## 123  6   48 27  B   3
```

The model used in this project assumes that individual parameters depend on constant and/or time varying covariates. Furthermore, IOV is assumed for parameters ka and V:

```
[COVARIATE]
input = {C1, C2, C3, C4, sC4, OCC, sOCC}
C3 = {type=categorical, categories={A, B}}
C4 = {type=categorical, categories={A, B}}
sC4 = {type=categorical, categories={A, A_B, A_B_B}}
OCC = {type=categorical, categories={1, 2, 3}}
sOCC = {type=categorical, categories={1, 1_2, 1_2_3}}

[INDIVIDUAL]
input = {ka_pop, beta_ka_C1, beta_ka_C2, beta_ka_C4_B, C1, C2, C4, omega_ka, gamma_ka,
  V_pop, beta_V_C2, beta_V_OCC_2, beta_V_OCC_3, OCC, omega_V, gamma_V, Cl_pop, beta_Cl_C1,
  beta_Cl_C3_B, C3, omega_Cl}
C4 = {type=categorical, categories={A, B}}
OCC = {type=categorical, categories={1, 2, 3}}
C3 = {type=categorical, categories={A, B}}

DEFINITION:
ka = {distribution=lognormal, typical=ka_pop, covariate={C1, C2, C4},
      coefficient={beta_ka_C1, beta_ka_C2, {0, beta_ka_C4_B}},
      varlevel={id, id*occ}, sd={omega_ka, gamma_ka}}
V = {distribution=lognormal, typical=V_pop, covariate={C2, OCC},
     coefficient={beta_V_C2, {0, beta_V_OCC_2, beta_V_OCC_3}},
     varlevel={id, id*occ}, sd={omega_V, gamma_V}}
Cl = {distribution=lognormal, typical=Cl_pop, covariate={C1, C3},
      coefficient={beta_Cl_C1, {0, beta_Cl_C3_B}}, sd=omega_Cl}

[LONGITUDINAL]
input = {a, b}

file = 'lib:oral1_1cpt_kaVCl.txt'

DEFINITION:
y1 = {distribution=normal, prediction=Cc, errorModel=combined1(a, b)}
```

New PK data can be simulated using this model with the Monolix project:

```
res <- simulx(project=iov.project, output= list(name=c("ka","V","Cl","OCC")))
names(res)
```

```
## [1] "y1"            "occasion"      "treatment"     "originalId"    "parameter.iiv"
## [6] "parameter.iov" "population"
```

```
print(head(res$parameter.iiv))
```

```
##   id         Cl       ka0         V0
## 1  1 0.01289110 0.5232405 0.16764302
## 2  2 0.02918496 0.4062680 0.21425064
```

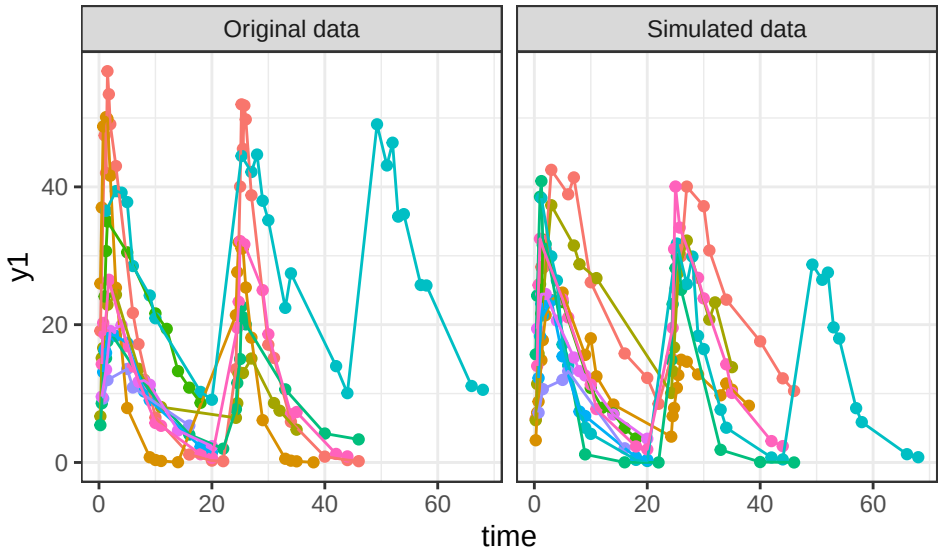
```
## 3  3 0.01733110 0.3623415 0.14038117
## 4  4 0.03293744 1.1094567 0.21924345
## 5  5 0.03974216 0.9165501 0.07372223
## 6  6 0.04355428 1.0197031 0.16219050
```

```
print(head(res$parameter.iov))
```

```
##    id time occ OCC        ka         V
## 1   1    0   1   2 0.5477658 0.1370929
## 2   1   24   2   3 0.7001230 0.1738018
## 27  2    0   1   2 0.3573879 0.1801452
## 28  2   24   2   3 0.3572564 0.2187890
## 52  3    0   1   2 0.3459839 0.1396220
## 53  3   24   2   3 0.4239869 0.1358902
```

We can then plot both the original and the simulated PK data:

```
y1 <- subset(d$y1, id %in% 1:10)
y1$out <- "Original data"
y2 <- subset(res$y1, id %in% 1:10)
y2$out <- "Simulated data"
y <- rbind(y1, y2)

pl <- ggplot(data=y, aes(x=time, y=y1)) + geom_point( aes(colour=id)) +
  geom_line( aes(colour=id)) + facet_wrap(~out) + theme(legend.position="none")

print(pl)
```



Gene expression in single cells

Introduction

The population approach may also be relevant for building predictive computational models of intracellular processes.

For example, consider as a case study the experiments performed by Uhlendorf *et al.*, 2012, looking at the high-osmolarity glycerol (HOG) pathway in budding yeast.

Yeast cells are exposed to osmotic shocks, i.e., sudden changes in the solute concentration of their surroundings. Signal transduction pathways, most notably the HOG pathway, provide information to the cell about the osmolarity of its environment and activate responses to deal with these stress conditions. In particular, a large set of genes is turned on and corresponding stress-responsive proteins are produced.

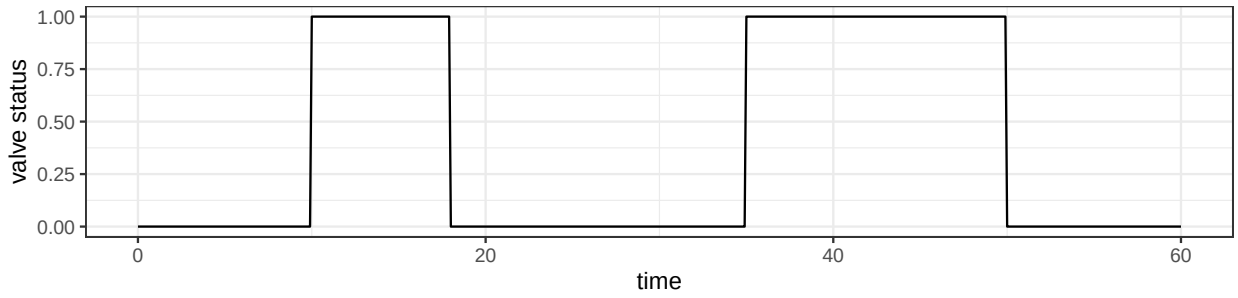
A model for the promoter activation rate

Let $v(t)$ be the valve status at time t with $v = 1$ when the valve is "on" (hyperosmotic media) and $v = 0$ when it is "off" (normal media).

Then, v is a piecewise constant function, solution of $\dot{v}(t) = 0$ with source terms equal to 1 when the valve is turned on, and equal to -1 when it is turned off.

In the following example, the valve is "on" between 10' and 18' and "off" betwen 35' and 50':

```
valveModel <- inlineModel("
[LONGITUDINAL]
PK:
depot(target=v)
EQUATION:
ddt_v=0
")

ton  <- list(amount = 1,  time   = c(10, 35))
toff <- list(amount = -1, time   = c(18, 50))
out  <- list(name = 'v', time = seq(0, 60, by=0.1))

res <- simulx(model=valveModel, output=out, treatment=list(ton,toff))
print(ggplot(res$v) + geom_line(aes(time,v)) + ylab('valve status'))
```



For modeling, we need first to take into account the lag between change in valve status and change of osmolarity $h(t)$ of the cell environment: We assume that the medium needs one minute to change from normal to hyperosmotic when the valve is turned on and four minutes to return to normal when turned off.

The function h can then be defined by introducing inputs of value 1 at times $(t_k^{on}, 1 \leq k \leq K)$ with a rate of 1, and inputs with value -1 at times $(t_k^{off}, 1 \leq k \leq K)$ with rate 0.25. Here, $t_k^{off} - t_k^{on}$ is the duration of the kth shock.

Then, the extracted nuclear Hog1 function $s(t)$ for a given salt concentration is a solution to the following ODE:

$$\dot{s}(t) = \kappa\, h(t) - \gamma\, s(t).$$

Lastly, the corresponding time-varying gene activation intensity $u(t)$ is obtained by transforming the Hog1 abundance using a Hill function:

$$u(t) = \frac{(s(t)+s_0)^{n_H}}{K_d^{n_H} + (s(t)+s_0)^{n_H}}.$$

This model is implemented in `hog1_model`.

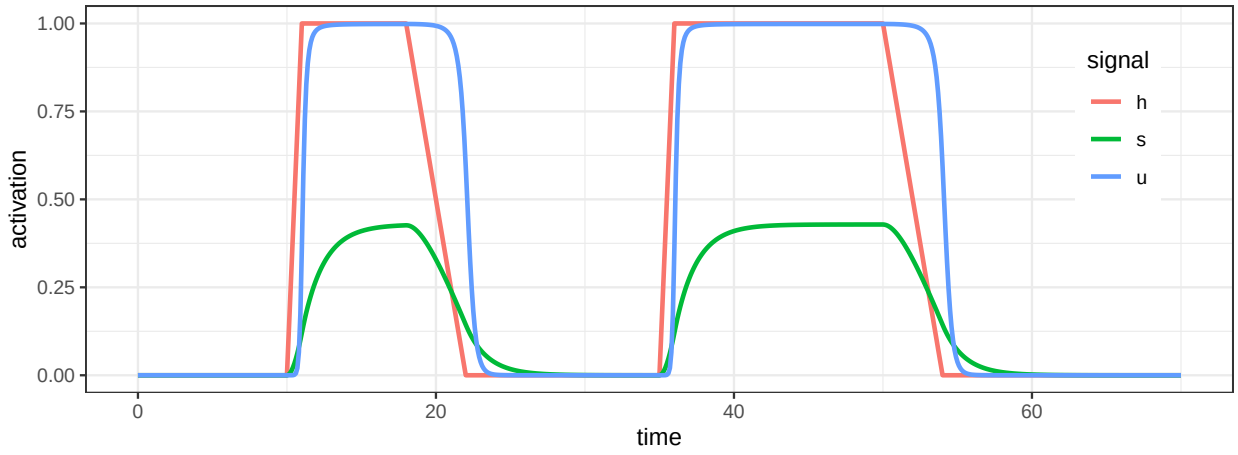
```
hog1_model <- inlineModel("
[LONGITUDINAL]
input={kappa, gamma, nh, Kd, s0}

PK:
depot(target=h)

EQUATION:
ddt_h = 0
ddt_s = kappa*h - gamma*s
sh    = (s+s0)^nh
u     = sh/(Kd^nh+sh)
")
```

Assume again that the valve is "on" between 10' and 18' and "off" betwen 35' and 50'. Status of the valve will be defined according to input source terms defined in lists `ton` and `toff`:

```
ton  <- list(amount = 1,
             rate   = 1,
             time   = c(10, 35))

toff <- list(amount = -1,
             rate   = 0.25,
             time   = c(18, 50))
```

We will display h, s and u computed between 0' and 70' with a given set of parameter values:

```
library("reshape2")

out  <- list(name = c('h','s','u'),
             time = seq(0, 70, by=0.1))

p1    <- c(kappa=0.3, gamma=0.7, nh=6, Kd=0.15, s0=0.016)

res <- simulx(model     = hog1_model,
              parameter = p1,
              output    = out,
              treatment = list(ton, toff))

r <- merge(res$h,merge(res$s,res$u))
r <- melt(r, id = 'time', variable.name = 'signal')
print(ggplot(r, aes(time,value)) + geom_line(aes(colour = signal),size=1) +
  ylab('activation')+ theme(legend.position=c(.9, .7)))
```



The Hog1-induced gene expression model

The related Hog1-induced gene expression model is given by a reaction network described by Zechner *et al.* (2012) and adapted by Gonzalez *et al.* (2013).

Let x_1, x_2 and x_4 be the proportions of promoters that are in the *off* state, in the *on* state or bound to chromatin remodeling factors. Let x_3, x_5 and x_6 be the concentrations of remodeling factors, messenger RNA and fluorescent yECitrine proteins.

The effective activation rate of the gene *pSTL1* is assumed to be proportional to the activation intensity u defined in the previous section with a delay τ.

Then, the system can be described by the following set of reaction rate equations:

$$
\begin{aligned}
\dot{x_1}(t) &= c_2 x_2(t) - c_1 u(t-\tau) x_1(t) \\
\dot{x_2}(t) &= c_1 u(t-\tau) x_1(t) - c_2 x_2(t) + c_4 x_4(t) - c_3 x_2(t) x_3(t) \\
\dot{x_3}(t) &= c_4 x_4(t) - c_3 x_2(t) x_3(t) \\
\dot{x_4}(t) &= c_3 x_2(t) x_3(t) - c_4 x_4(t) \\
\dot{x_5}(t) &= c_5 x_4(t) - c_8 x_5(t) \\
\dot{x_6}(t) &= c_6 x_5(t) - c_7 x_6(t),
\end{aligned}
$$

where the initial conditions at $t = t_0$ are $x_1(t_0) = 1$, $x_2(t_0) = x_4(t_0) = x_5(t_0) = x_6(t_0) = 0$ and $x_3(t_0) = x_{3,0}$.

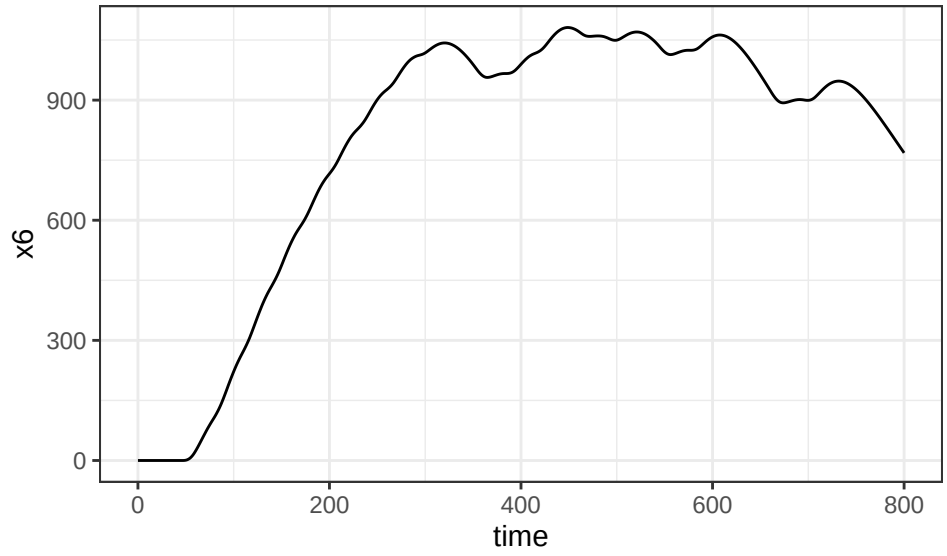
The structural model combines this dynamical system with the model for the activation intensity described in the previous section. It is implemented in `hog2_model`:

```
hog2_model <- inlineModel("
[LONGITUDINAL]
input={kappa, gamma, nh, Kd, s0, c1, c2, c3, c4, c5, c6, c7, c8, x30, tau}

PK:
depot(target=h,Tlag=tau)

EQUATION:
ddt_h = 0
ddt_s = kappa*h - gamma*s
sh    = (s+s0)^nh
u     = sh/(Kd^nh+sh)

x1_0 = 1
```

```
x3_0 = x30
ddt_x1 = c2*x2 - c1*u*x1
ddt_x2 = c1*u*x1 - c2*x2 + c4*x4 - c3*x2*x3
ddt_x3 = c4*x4 - c3*x2*x3
ddt_x4 = c3*x2*x3 - c4*x4
ddt_x5 = c5*x4 - c8*x5
ddt_x6 = c6*x5 - c7*x6
")
```

Let us use `simulx` for computing x_6 using the experimental design used by Gonzalez *et al.* (2012)

```
ton  <- list(amount = 1,
             rate   = 1,
             time   = c(35, 65, 95, 125, 155, 185, 215, 245, 280, 341, 371, 401,
                        449, 479, 533, 563, 649, 683))

toff <- list(amount = -1,
             rate   = 0.25,
             time   = c(43, 73, 103, 133, 163, 193, 223, 253, 285, 349, 379, 409,
                        454, 486, 541, 570, 657, 691))

out  <- list(name=c('x6'),  time=seq(0, 800, by=0.5))

p2 <- c(c1=75, c2=2500, c3=0.00002, c4=0.04, c5=1.2, c6=800, c7=0.005, c8=0.1, x30=150, tau=10)

res <- simulx(model     = hog2_model,
              parameter = c(p1, p2),
              output    = out,
              treatment = list(ton, toff))

plot(ggplot() + geom_line(data=res$x6, aes(x=time, y=x6)))
```



1. Gonzalez, A., Uhlendorf, J., Schaul, J., Cinquemani, E., Batt, G., and Ferrari-Trecate, G. (2013). Identification of biological models from single-cell data: a comparison between mixed-effects and

momentbased inference. In Proceedings of ECC - 12th European Control Conference, pages 2652-2657.

2. Uhlendorf, J., Miermont, A., Delaveau, T., Charvin, G., Fages, F., Bottani, S., Batt, G., and Hersen, P. (2012). Long-term model predictive control of gene expression at the population and single-cell levels. Proceedings of the National Academy of Sciences, 109(35):14271-14276.

3. Zechner, C., Ruess, J., Krenn, P., Pelet, S., Peter, M., Lygeros, J., and Koeppl, H. (2012). Moment-based inference predicts bimodality in transient gene expression. Proceedings of the National Academy of Sciences, 109(21):8340-8345.

Some tumor growth models

Delbene

Mathematical model:

$$\begin{aligned}
\dot{N}_1(t) &= k_2\, C\, N_p(t) - k_1\, N_1(t) \\
\dot{N}_2(t) &= k_1\, N_1(t) - k_1\, N_2(t) \\
\dot{N}_3(t) &= k_1\, N_2(t) - k_1\, N_3(t) \\
\dot{N}_p(t) &= \lambda_0\, N_p(t) - k_2\, C\, N_p(t) \\
N_t(t) &= N_1(t) + N_2(t) + N_3(t) + N_p(t)
\end{aligned}$$

Initial conditions:

$$\begin{aligned}
N_1(0) = N_2(0) = N_3(0) &= 0 \\
N_p(0) &= N_0
\end{aligned}$$

Del Bene F, Germani M, De Nicolao G, Magni P, Re CE, Ballinari D, Rocchetti M *A model-based approach to the in vitro evaluation of anticancer activity*, Cancer chemotherapy and pharmacology, 4/2009, Volume 63, Issue 5, pages: 827-836

Ribba

Mathematical model:

$$\begin{aligned}
P^\star(t) &= P_T(t) + Q(t) + Q_P(t) \\
\dot{C}(t) &= -k_{de} C(t) \\
\dot{P_T}(t) &= \lambda P_T(t)(1 - P^\star(t)/K) + k_{QPP} Q_P(t) - k_{PQ} P_T(t) - \gamma\, k_{de} P_T(t) C(t) \\
\dot{Q}(t) &= k_{PK} P_T(t) - \gamma\, k_{de} Q(t) C(t) \\
\dot{Q_P}(t) &= \gamma\, k_{de} Q(t) C(t) - k_{QPP} Q_P(t) - \delta_{QP} Q_P(t)
\end{aligned}$$

Initial conditions:

$$\begin{aligned}
C(0) = Q_P(0) &= 0 \\
P_T(0) &= p_{T0} \\
Q(0) &= q_0
\end{aligned}$$

Treatment:

- administration $\Longrightarrow C$

Ribba, B., Kaloshi, G., Peyre, M., Ricard, D., Calvez, V., Tod, M., ... & Ducray, F. (2012). *A tumor growth inhibition model for low-grade glioma treated with chemotherapy or radiotherapy.* Clinical Cancer Research, 18(18), 5071-5080.

Simeoni

Mathematical model:

$$
\begin{aligned}
C &= Q_1/V_1 \\
\dot{Q_1}(t) &= k_{21}Q_2(t) - (k_{10} + k_{12})Q_1(t) \\
\dot{Q_2}(t) &= k_{12}Q_1(t) - k_{21}Q_2(t) \\
\dot{Z_0}(t) &= \frac{\lambda_0 Z_0(t)}{\left(1 + (W\,\lambda_0/\lambda_1)^{\psi}\right)^{\frac{1}{\psi}}} - k_2 C\, Z_0 \\
\dot{Z_1}(t) &= k_2 C\, Z_0(t) - k_1 Z_1(t) \\
\dot{Z_2}(t) &= k_1 Z_1(t) - k_1 Z_2(t) \\
\dot{Z_3}(t) &= k_1 Z_2(t) - k_1 Z_3(t) \\
W(t) &= Z_0(t) + Z_1(t) + Z_2(t) + Z_3(t)
\end{aligned}
$$

Initial conditions:

$$
\begin{aligned}
Q_1(0) = Q_2(0) = Z_1(0) = Z_2(0) = Z_3(0) = 0 \\
Z_0(0) = w_0
\end{aligned}
$$

Treatment:

- administration $\Longrightarrow Q_1$

  Simeoni M, Magni P, Cammia C, De Nicolao G, Croci V, Pesenti E, Germani M, Poggesi I, Rocchetti, *Predictive pharmacokinetic-pharmacodynamic modeling of tumor growth kinetics in xenograft models after administration of anticancer agents.* M Cancer research, 2/2004, Volume 64, Issue 3, pages: 1094-1101

Simeoni revisited

Simeoni *et al.* (2004) presented a model for tumor growth and anti-cancer effects where a transit compartment model describes the apoptosis of tumor cells attacked by a drug. Following Koch et al. (2014), this model can be rewritten with a lifespan model.

Mathematical model:

$$
\begin{aligned}
\dot{A_d}(t) &= -k_a\, A_d(t) \\
\dot{A_c}(t) &= k_a\, A_d(t) - k_e\, A_c(t) \\
C_c(t) &= A_c(t)/V \\
\dot{p}(t) &= \frac{2 l_0 l_1 p(t)^2}{(l_1 + 2 l_0 p(t))(p(t) + d(t))} - k_{\text{pot}} C_c(t) p(t) \\
\dot{d}(t) &= k_{\text{pot}} C_c(t) p(t) - k_{\text{pot}} p(t-\tau) \frac{A_c(t-\tau)}{V} \\
w(t) &= p(t) + d(t)
\end{aligned}
$$

Initial conditions:

$$
\begin{aligned}
A_c(t) &= 0 \\
p(t) &= w_0 \\
d(t) &= 0
\end{aligned}
$$

The PK parameters (k_a, V, k_e) are fixed:

$$
k_a = 20 \quad ; \quad V = 2.8 \quad ; \quad k_e = 2.4
$$

Treatment:

- administration $\Longrightarrow A_d$

  Koch G, Krzyzanski W, Perez-Ruixo JJ, Schropp J. (2014) *Modeling of delays in PKPD: classical approaches and a tutorial for delay differential equations*, J Pharmacokinet Pharmacodyn. 41(4):291-318.

Rocchetti

Mathematical model:

$$
\begin{aligned}
c_{1A} &= FV_{1A}Q_{1A} \\
c_{1B} &= FV_{1B}Q_{1B} \\
k_{2h} &= k_2\left(1 - \frac{c_{1A}}{IC_{50c} + c_{1A}}\right) \\
\dot{Q_{0A}}(t) &= -k_{aA}Q_{0A}(t) \\
\dot{Q_{1A}}(t) &= k_{aA}Q_{0A}(t) - k_{eA}Q_{1A}(t) \\
\dot{Q_{0B}}(t) &= -k_{aB}Q_{0B}(t) \\
\dot{Q_{1B}}(t) &= k_{aB}Q_{0B}(t) - (k_{12} + k_{eB})Q_{1B}(t) - k_{21}Q_{2B}(t) \\
\dot{Q_{2B}}(t) &= k_{12}Q_{1B}(t) - k_{21}Q_{2B}(t) \\
\dot{Z}_0(t) &= \frac{\lambda_0 Z_0(t)}{\left(1 + (W\,\lambda_0/\lambda_1)^{\psi}\right)^{\frac{1}{\psi}}}\left(1 - E_{\max}\frac{c_{1A}}{IC_{50} + c_{1A}}\right) - k_{2h}c_{1B} \\
\dot{Z}_1(t) &= k_{2h}c_{1B}Z_0(t) - k_1Z_1(t) \\
\dot{Z}_2(t) &= k_1Z_1(t) - k_1Z_2(t) \\
\dot{Z}_3(t) &= k_1Z_2(t) - k_1Z_3(t) \\
W(t) &= Z_0(t) + Z_1(t) + Z_2(t) + Z_3(t)
\end{aligned}
$$

Initial conditions:

$$
\begin{aligned}
Q_{0A}(0) = Q_{1A}(0) = Q_{0B}(0) = Q_{1B}(0) = Q_{2B}(0) &= 0 \\
Z_0(0) &= w_0 \\
Z_1(0) = Z_2(0) = Z_3(0) &= 0
\end{aligned}
$$

Treatment:

- administration of *type 1* $\Longrightarrow Q_{0A}$
- administration of *type 2* $\Longrightarrow Q_{1B}$

  Rocchetti M, Germani M, Del Bene F, Poggesi I, Magni P, Pesenti E, De Nicolao G, *Predictive pharmacokinetic-pharmacodynamic modeling of tumor growth after administration of an anti-angiogenic agent, bevacizumab, as single-agent and combination therapy in tumor xenografts*, Cancer chemotherapy and pharmacology, 5/2013, Volume 71, Issue 5, pages: 1147-1157