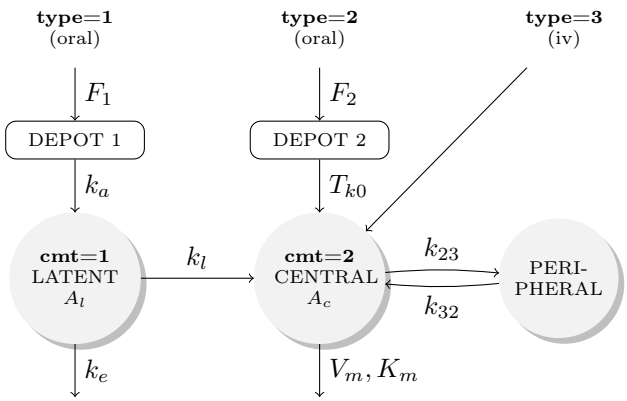
1 Complex PK model

In this example,

- one type of dose is administered orally (`type=1`) and absorbed into a latent compartment following a first-order absorption process,
- a second type is administered orally (`type=2`) and absorbed into the central compartment following a zero-order absorption process,
- a third type is directly administered intravenously to the central compartment (`type=3`).

The transfer from the latent to the central compartment is linear. A peripheral compartment is linked to the central compartment. The drug is eliminated by a linear process from the latent compartment and a nonlinear process from the central one. Here, A_l and A_c are the drug amounts in the latent and central compartments.



Implementation of this model using PK macros:

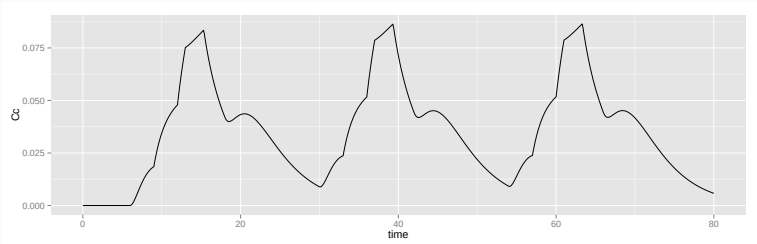
pk_model.txt

```
[LONGITUDINAL]
input = {F1, F2, ka, Tk0, kl, k23, k32, V, k, Vm, Km}

PK:
compartment(cmt=1, amount=Al)
compartment(cmt=2, amount=Ac)
peripheral(k23,k32)
oral(type=1, cmt=1, ka,  p=F1)
oral(type=2, cmt=2, Tk0, p=F2)
iv(type=3, cmt=2)
transfer(from=1, to=2, kt=kl)
elimination(cmt=1, k)
elimination(cmt=2, Km, Vm)
Cc = Ac/V
```

Example of R script for computing the concentration in the central compartment using `simulx`:

pk_simulx.R

```
adm1 <- list( time   = seq(6, 66, by=12),
              amount = 2,
              type   = 1)

adm2 <- list( time   = seq(9, 57, by=24),
              amount = 1,
              type   = 2)

adm3 <- list( time   = seq(12, 60, by=24),
              amount = 1,
              rate   = 0.3,
              type   = 3)

p <- list( name  = c('F1','F2','ka','Tk0','kl','k23','k32','V','k','Vm','Km'),
           value = c(0.5, 0.8, 0.5, 4, 0.5, 0.3, 0.5, 10, 0.2, 1, 2))

Cc <- list( name = "Cc",
            time = seq(from=0,to=80,by=0.1))

res <- simulx( model     = "pk_model.txt",
               parameter = p,
               output    = Cc,
               treatment = list(adm1, adm2, adm3))

plot1 <- ggplot(data=res$Cc, aes(x=time, y=Cc, group=id)) + geom_line()

> print(plot1)
```



2 Hierarchical model

Mixed effects models are hierarchical models that involves different types of variables:

- We call $y_i = (y_{ij}, 1 \leq j \leq n_i)$ the set of *longitudinal data* recorded at times $(t_{ij}, 1 \leq j \leq n_i)$ for subject i, and $\boldsymbol{y}$ the combined set of observations for all N individuals: $\boldsymbol{y} = (y_1, \ldots, y_N)$.
- We write ψ_i for the vector of *individual parameters* for individual i and $\boldsymbol{\psi}$ the set of individual parameters for all N individuals: $\boldsymbol{\psi} = (\psi_1, \ldots, \psi_N)$.
- The distribution of the individual parameters ψ_i of subject i may depend on a vector of *individual covariates* c_i.
- In a population approach context, we call θ the vector of *population parameters*.

Considering these variables as random variables, the joint probability distribution of $\boldsymbol{y}$, $\boldsymbol{\psi}$, $\boldsymbol{c}$ and θ can be decomposed into a product of conditional distributions:

$$\mathrm{p}(\boldsymbol{y}, \boldsymbol{\psi}, \boldsymbol{c}, \theta) = \mathrm{p}(\boldsymbol{y}|\boldsymbol{\psi}, \theta)\mathrm{p}(\boldsymbol{\psi}|\boldsymbol{c}, \theta)\mathrm{p}(\boldsymbol{c}|\theta)\mathrm{p}(\theta)$$

MLXTRAN takes advantage of the hierarchical structure of this joint probability distribution by decomposing the joint model into several submodels.

The example below shows how each component of the model is implemented in a different section. Each section combines equations and/or definitions.

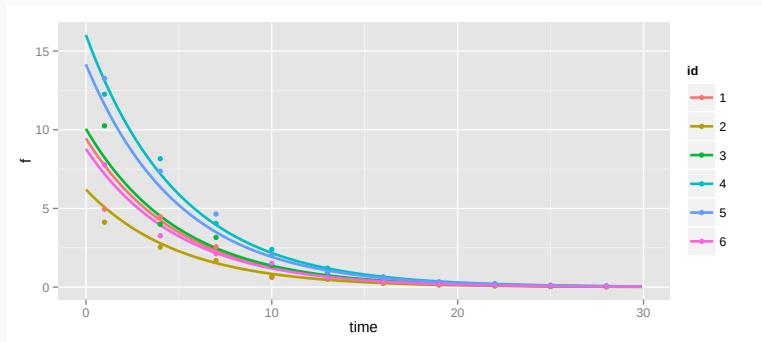
hierarchical_model.txt

```
[LONGITUDINAL]
input = {V, k, b}
EQUATION:
D=100
f = D/V*exp(-k*t)
DEFINITION:
y = {distribution=normal, prediction=f, sd=b*f}

[INDIVIDUAL]
input = {V_pop, omega_V, w, w_pop}
EQUATION:
V_pred = V_pop*(w/w_pop)
DEFINITION:
V = {distribution=logNormal, prediction=V_pred, sd=omega_V}

[COVARIATE]
input = {w_pop, omega_w}
DEFINITION:
w = {distribution=normal, mean=w_pop, sd=omega_w}

[POPULATION]
input = {ws, gw, Vs, gV}
DEFINITION:
w_pop = {distribution=normal, mean=ws, sd=gw}
V_pop = {distribution=logNormal, mean=log(Vs), sd=gV}
```

The `simulx` script now includes the values of the *hyperparameters* that define the distribution of the population parameters. In this example, 2 populations and 3 individuals in each population with their own covariates are simulated:

hierarchical_simulx.R

```
p <- list( name  = c('ws','gw','Vs','gV','omega_w','omega_V','k','b'),
           value = c(70,10,10,0.1,12,0.15,0.2,0.2))

f <- list( name = 'f',
           time = seq(0, 30, by=0.1))

y <- list( name = 'y',
           time = seq(1, 30, by=3))

ind <- list(name=c('w','V'))
pop <- list(name=c('w_pop','V_pop'))
out <- list(pop, ind, f, y)

g <- list(size=c(2,3),
          level=c('population','covariate'))

res <- simulx( model     = 'hierarchical_model.txt',
               parameter = p,
               output    = out,
               group     = g)


plot1 <- ggplot(data=res$f, aes(x=time, y=f, colour=id)) +
  geom_line(size=1) +
  geom_point(data=res$y, aes(x=time, y=y, colour=id))

> print(plot1)
```




```
> print(res$parameter)

  id    w_pop    V_pop        w         V
1  1 78.43234 9.991236 72.52128 10.585047
2  2 78.43234 9.991236 77.44056 16.127633
3  3 78.43234 9.991236 85.61637  9.945883
4  4 76.06347 8.817725 69.23987  6.239125
5  5 76.06347 8.817725 81.63738  7.063361
6  6 76.06347 8.817725 82.68483 11.386175
```

3 Multiple outputs model

Joint model for PK, continuous PD and categorical PD data:

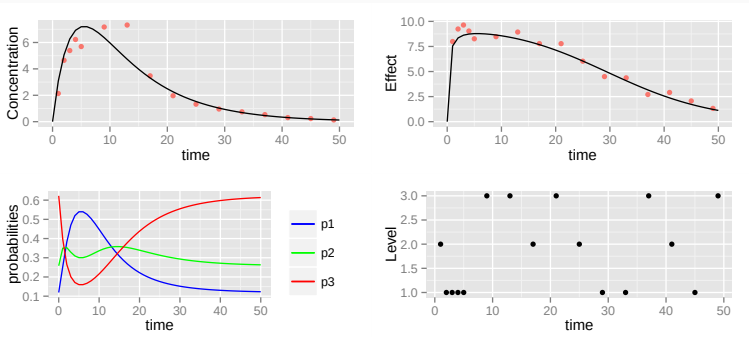
joint_model.txt

```
[LONGITUDINAL]
input = {ka, V, k, a1, a2, Emax, EC50, th1, th2, th3}

PK:
depot(target=Ad)

EQUATION:
ddt_Ad = -ka*Ad
ddt_Ac =  ka*Ad - k*Ac
Cc = Ac/V
E = Emax*Cc/(EC50 + Cc)
lp1 = th1 + th2*Cc
lp2 = th1 + th2*Cc + th3


DEFINITION:
Concentration = {distribution=lognormal, prediction=Cc, sd=a1}

Effect = {distribution=normal, prediction=E, sd=a2}

Level = {type=categorical, categories={1,2,3},
               logit(P(Level<=1)) = lp1
         logit(P(Level<=2)) = lp2 }
```

We use `simulx` for computing quantities and simulating data from this model.

joint_simulx.R

```
a <- list(time   = 0,
          amount = 100)

p <- list(name=c('ka','V','k','a1','Emax','EC50','a2','th1','th2','th3'),
          value=c(0.3,8,0.1,0.2,10,1,0.5,-2,0.3,1.5))

f <- list(name=c('Cc', 'E', 'lp1', 'lp2'),
          time=seq(0, 50, by=1))
y <- list(name=c('Concentration','Effect','Level'),
          time=c(1,2,3,4,seq(5,50,by=4)))

res <- simulx(model='joint_model.txt',
              treatment=a,
              parameter=p,
              output=list(f, y));

n=length(f$time)
p1=1/(1+exp(-res$lp1$lp1[1:n]))
p2=1/(1+exp(-res$lp2$lp2[1:n])) -p1
proba <- data.frame(time=res$lp1$time[1:n],p1=p1,p2=p2,p3=1-p1-p2)
res$Cc=res$Cc[1:n,]
res$E=res$E[1:n,]

plot1=ggplot() + geom_point(data=res$Concentration,
  aes(x=time, y=Concentration, colour=id)) +
  geom_line(data=res$Cc, aes(x=time, y=Cc))  +
  theme(legend.position="none")

plot2=ggplot() + geom_point(data=res$Effect,
  aes(x=time, y=Effect, colour=id)) +
  geom_line(data=res$E, aes(x=time, y=E))  +
  theme(legend.position="none")

plot3=ggplot() +
  geom_line(data=proba, aes(x=time, y=p1, colour="blue"))  +
  geom_line(data=proba, aes(x=time, y=p2, colour="green"))  +
  geom_line(data=proba, aes(x=time, y=p3, colour="red"))+
  scale_colour_manual(name="",
  values=c('green'='green','blue'='blue','red'='red'),
  labels=c('p1','p2','p3')) +
  ylab("probabilities")
```



4 Time-to-event data model

The distribution of time-to-event data (or *survival* data) is defined with the *hazard* function. Some additional information should be provided in the model (maximum number of events, right censoring time, interval censoring, ...).

We consider a Weibull model for a single event in this example, with a right censoring time equal to 100.

event1_model.txt

```
[LONGITUDINAL]
input = {beta,lambda}

EQUATION:
h=(beta/lambda)*(t/lambda)^(beta-1)

DEFINITION:
e = {type=event, maxEventNumber=1, rightCensoringTime=100, hazard=h}
```

The following model uses the same hazard function, assuming now that the events are repeated and interval censored:

event2_model.txt

```
[LONGITUDINAL]
input = {beta,lambda}

EQUATION:
h=(beta/lambda)*(t/lambda)^(beta-1)

DEFINITION:
e = {type=event, eventType=intervalCensored,
     intervalLength=10, rightCensoringTime=100, hazard=h}
```

We can use `simulx` for simulating 100 individuals with these two models. It is mandatory to state explicitly when the experiment starts (at time $t = 0$ in this example):

event1_simulx.R

```
p <- list( name = c('beta','lambda'), value=c(2.5,50))
h <- list(name='h', time=seq(from=0, to=30, by=0.1))
e <- list(name='e', time=0)
o <- list(h, e);
g <- list(size=100)
res1 <- simulx(model='event1_model.txt',parameter=p,output=o,group=g)
res2 <- simulx(model='event2_model.txt',parameter=p,output=o,group=g)
```