METRUM
RESEARCH GROUP

mrgsolve: Model Specification

mrgsolve Workshop
March 12, 2016
San Diego, CA

Available code blocks

- $PARAM, $THETA, $FIXED
- $CMT, $INIT, $VCMT
- $MAIN
- $ODE
- $TABLE , $CAPTURE
- $OMEGA , $SIGMA
- $GLOBAL
- $PROB, $SET, $ADVAN2, $ADVAN4, $NMXML, $CMTN

Assemble blocks into something like this

```
$PARAM CL=1, VC=35, VMAX=2, KM=1.2 KA=1.1

$CMT SC CENT

$MAIN
double VCi = VC*exp(ETA(2));

$OMEGA 0.5 0.5

$ODE
dxdt_SC =  -KA*SC;
dxdt_CENT = KA*SC - (CL/VCi)*CENT;

$TABLE table(CP) = CENT/VCi; // forget about this
```

$PARAM [R]

- Model “parameters” with name=value format
  - Separate by , or \n
  - Use any name except for words in mrgsolve::reserved()
  - Values are evaluated by R parser
- Multiple blocks allowed
- Use name anywhere in the model
- Updatable in R but not in the model spec
  - Most often, you will want to match names in $PARAM with the names in your input data sets

```
$PARAM
TVCL=1.1, TVVC=1.2
KA = 0.7, F1 = sqrt(2/3)
```

- See also: $THETA and $FIXED
- **ACCESS** R global environment

$CMT [txt] and $INIT [R]

- Specify name and number of compartments in the model
- Use any name except for those listed in `mrgsolve::reserved()`
- Use either `$CMT` or `$INIT`
- Maybe used to specify initial condition
  - `$CMT` specify compartment names only
  - `$INIT` to specify `name` = `<initial>`

See also: `$VCMT`

ACCESS `R` global environment for `$INIT` only

The difference between $CMT and $INIT

- These give identical results

```
$CMT GUT CENT RESP
```

```
$INIT GUT=0, CENT=0, RESP=0
```

- But you may wish to do this

```
$INIT GUT=0, CENT=0, RESP=50
```

Variables and macros (1/3)

- CMT_0 the initial condition for CMT
- CMT the current amount in CMT

- TIME the current data set time
- SOLVERTIME in $ODE only ... the current ode solver time

- EVID event id
- ID subject ID
- NEWIND new individual flag
  - 0 for first record of data set
  - 1 for first record of this ID
  - 2 for subsequent of this ID

Variables and macros (2/3)

- Random effects
  - ETA(m)
  - EPS(m)
- General dosing
  - _F(n) bioavailability parameter
  - _ALAG(n) dose lag time
- Infusion settings
  - _D(n) used to set infusion duration
  - _R(n) used to set infusion rate

n (compartment number) and m (ETA or EPS number) are integers

Variables and macros (3/3)

- `_NEQ` number of equations
- `_N_CMT` the number (integer) of compartment `CMT`; but **only** if you used `$CMTN`
- `$ADVAN2`
  - `pred_CL`, `pred_VC`, `pred_KA`
- `$ADVAN4`
  - `pred_CL`, `pred_VC`, `pred_KA`, `pred_Q`, `pred_VP`
- Synonyms
  - `pred_VC`, `pred_V2` and `pred_V` are synonymous for central volume
  - `pred_VP` and `pred_V3` are synonymous for peripheral volume

$MAIN [C++]

- Like $PK, code that gets called right before solving odes from t_1 to t_2
- Set initial conditions
- Derive new variables . . . includes covariate modeling
  - Must type new variables . . . usually double or bool
  - By default, new variables are initialized globally

```
$MAIN
RESP_0 = KIN/KOUT;

double CLi = exp(log(TVCL) + ETA(1));

double base = 0;

if(NEWIND <= 1) base = 2;
```

ACCESS parameters, compartment amounts, compartment initial conditions all variables and macros

$ODE [C++]

- Every compartment **needs** an equation
- Form: `dxdt_CMT = ratein - rateout;`
- Use `CMT` for compartment amounts, parameters, or other variables
- This block of code gets called repeatedly so be wise
- May specifiy new variables here as in `$MAIN`; they will be initialized globally

```
$ODE
dxdt_CENT  = -(CL/VC)*CENT;

double CP = CENT/VC;
dxdt_RESP = KIN*(1-CP/(IC50+CP)) - KOUT;
```

$TABLE [C++] and $CAPTURE [txt]

- Used to derive variables **after** the solver advances the system
- Specify additional simulation outputs
  - Derived variables
  - Internal variables (like `ETA(n)` or `EPS(n)`)
  - Parameters
- Do not table compartments or `ID` or `TIME`
- `$CAPTURE` is convenience wrapper for table statement

```
$TABLE
table(logRESP) = log(RESP);
```

$CAPTURE

- Items in $CAPTURE get translated to `table(name) = name;` in $TABLE

```
$PARAM FLAG=2, TVCL=1, VC=12
$CMT CENT
$MAIN double CLi = TVCL*exp(ETA(1));
$ODE
double rateout = (CLi/VC)*CENT;
dxdt_CENT = -rateout;
$CAPTURE rateout CLi FLAG
```

- These are equivalent

```
$TABLE table(CP) = CP;
```

```
$CAPTURE CP
```

Example: Simple PK

```
$PROB
PK base model development started 3/12/2016
Kyle Baron

$PARAM CL=1, VC=30

$CMT CENT

$ODE
dxdt_CENT = -(CL/VC)*CENT;

$TABLE table(CP) = CENT/VC;
```

Example: IRM with reduced bioavability

```
$PARAM CL=1, VC=30, KA=1.1
KIN=100, KOUT=2, IC50 = 2, FBIO=0.6

$CMT CENT RESP

$MAIN
RESP_0 = KIN/KOUT;
_F(1)  = FBIO;

$ODE
double CP = CENT/VC;
dxdt_CENT = -(CL/VC)*CENT;
dxdt_RESP = KIN*(1-CP/(IC50+CP)) - KOUT*RESP;

$CAPTURE FBIO
```

$OMEGA and $SIGMA [txt]

- Specify random effect variance/covariance matrices
  - $\eta_i \sim N(0, \Omega)$
  - $\varepsilon_j \sim N(0, \Sigma)$
- NONMEM-style input - lower triangle
- Multiple $OMEGA and $SIGMA are allowed and each may be named
  - How you enter these blocks defines a “signature” that mrgsolve will enforce later on
- Diagonal matrix (3x3 in this example)

```
$OMEGA 0.1 0.2 0.3
```

More `$OMEGA` and `$SIGMA`

- Options are `name=value` and must be on different line from the matrix data
- Block matrix (2x2 in this example)

```
$OMEGA block=TRUE
0.1 0.06 0.2
```

- Block matrix with correlations (assumes `block=TRUE`)

```
$OMEGA cor=TRUE
0.1 0.67 0.2
```

More $OMEGA and $SIGMA

- Multiple $OMEGA and $SIGMA are allowed; each may be named

```
$OMEGA name='PK'
0 0 0
$OMEGA name='PD'
0 0
```

Example: Population PK

```
$PARAM LTVCL=log(1), LTVVC=log(30), WT = 70, PROT=1

$CMT CENT

$MAIN
double CL = exp(LTVCL + 0.75*log(WT/70.0) +  ETA(1));
double VC = exp(LTVVC +     *log(WT/70.0) +  ETA(2));

$OMEGA cor=TRUE
0.2 0.67 0.2

$SIGMA 0.025 0.1

$ODE dxdt_CENT = -(CL/VC)*CENT;

$TABLE
double eps = PROT==1 ? EPS(1) : EPS(2);
table(DV) = (CENT/VC)*exp(eps);

$CAPTURE CL
```

©2016 Metrum Research Group LLC

C++ expressions and functions

```
if(a == 2) b = 2;

if(b <= 2) {
  c=3;
} else {
  c=4;
}

double d = pow(base,exponent);
double e = exp(3);
double f = fabs(-4);
double g = sqrt(5);
double h = log(6);
double i = log10(7);
double j = floor(4.2);
double k = ceil(4.2);
```

$GLOBAL [C++]

- Used to define C++ variables or data structures “globally”
    - Here, “globally” refers to the model shared object
    - Variables won’t clash with R or mrgsolve internals
- Also define C++ preprocessor directives here

```
$GLOBAL
bool cured = false;
double mult = 0;

#define g 9.8
#define plus(a,b) (a+b)
```

```
$GLOBAL
std::vector<double> a(5);
double mult(double a, double b) {return a*b;}
```

$SET [R]

- Define some settings
  - Simulation times: `start`, `end`, `delta`, `add`
  - Solver settings: `hmax`, `atol`, `rtol`, `maxsteps`
  - Compartment requests: `request`
  - Time re-scaling: `tscale`
- All of these settings can get changed later, but setting them here allows you bind some default settings to the specification file For example:

```
$SET
end=24*7, delta=1
hmax=0.1
tscale = 1/24
```

$THETA [txt]

```
$THETA
0.1 0.2 0.3
```

- Create items in the parameter list with name `THETAn`
- A quick way to get `THETA`-related parameters in to the list without a lot of typing
- Add a `name` argument to change the prefix

```
$THETA name="theta"
1 2 3
```

- Will insert `theta1=1`, `theta2=2` and `theta3=3` into parameter list

$THETA example

Data items in $THETA get inserted into $PARAM, but with an assumed name

```
code <- '
$THETA 8 9 10
$PARAM CL=4, VC=5
'
mread(code=code, "modspec", proj) %>% param
```

```
.
.  Model parameters (N=5):
.  name   value . name   value
.  CL     4     | THETA3 10
.  THETA1 8     | VC     5
.  THETA2 9     | .      .
```

$FIXED [R]

```
$FIXED
g = 9.8

$PARAM
TVCL = 5
```

- Functions just like `$PARAM`, but values are eternally fixed . . . even in R
- Declared as `const double`
- Usually only used when there is a large parameter list
  - Some performance improvement
  - Makes the parameter list smaller and easier to deal with
- **ACCESS** R global environment

$VCMT

- Use $VCMT to implement an indicator compartment
  - mrgsolve will write dxdt_FLAG=0;
  - You can dose into or use value in that compartment anywhere in the model
  - Just for convenience

```
$VCMT FLAG
```

$ADVAN2 [txt]

- Including this block tells mrgsolve to implement a one-compartment model with first-order absorption
- Must include $CMT with 2 compartments
  - First name is the dosing compartment and second is the central compartment
  - But you decide on the names

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

$CMT DEPOT CENT

$MAIN
double CLi = exp(log(CL) + ETA(1));

pred_CL = CLi;
pred_VC = VC;
pred_KA = KA;
```

$ADVAN4 [txt]

- Including this block tells mrgsolve to implement a two-compartment model with first-order absorption
- Must include CMT with 3 compartments
  - First name is dosing, second is central, third is peripheral
  - But you decide on the names

```
$PARAM CL=1, VC=20, KA=1.1
VP = 120 Q=5

$CMT DEPOT CENT PERIPH

$MAIN
double CLi = exp(log(CL) + ETA(1));

pred_CL = CLi;
pred_VC = VC;
pred_KA = KA;
```