Rationale for design of Pharsight Modeling Language (PML)

Michael R. Dunlavey, Robert H. Leary

Certara/Pharsight Corp.

Introduction

PML uses similar semantics to other languages for nonlinear mixed-effect modeling, with fixed effects, random effects, and observational error, combined with ordinary differential equations and arbitrary dosing and time-varying covariates. PML differs in that it tries to follow the philosophy of declarative languages, in which the language describes the problem, not the solution to the problem. To this end:

1. It simplifies the writing of unusual models, such as enterohepatic reflux, time-to-event, count, ordinal, bql, etc.
2. It is not necessary to rewrite the model to select a different ODE solver or a different modeling engine. It makes heavy use of symbolic differentiation to accomplish this.
3. It can be embedded in an existing statistics-oriented language (R), facilitating orchestration of model development from within scripts.
4. It facilitates trial simulation by combining fitted models, without the need to manually transcribe them (a process that introduces errors).

1. Simplify the writing of unusual models

```
# TIME TO EVENT
event(occur, hazard)

# COUNT
count(n, hazard)

# ORDINAL
ordinal(obs, linkfunc, x, slope, icept1, icept2, ...)

# BQL
observe(cObs = c * (1 + eps), bql)

# ENTEROHEPATIC REFLUX
sequence {
  reflux = 0  sleep( initialOffTime )
  while(1){ #### REPEAT CYCLE AS LONG AS NEEDED
    reflux = 1  sleep( onTime )
    reflux = 0  sleep( offTime )
  }
}
```

2. Symbolic differentiation allows ODE model to be used with Matrix-exponent, and stiff ODE solvers, obviates explicit Mu-modeling, etc.

```
# TWO COMPARTMENT MODEL WITH CLEARANCE
deriv(A1 = - Cl * C - Cl2 * (C - C2))
deriv(A2 = Cl2 * (C - C2))
C = A1 / V
C2 = A2 / V2
stparm(V = tvV * exp(nV), V2 = tvV2 * exp(nB2))
stparm(Cl = tvCl * exp(nCl), Cl2 = tvCl2 * exp(nCl2))

// GENERATED JACOBIAN C CODE
__temp[0] = exp(nCl);
__temp[1] = exp(nV);
__temp[2] = exp(nCl2);
__temp[3] = exp(nV2);
```

(continued…)

```
// rate terms for A1
  // w.r.t. A1
_jacobian[0 + _nrj*0] =
   (( -((tvCl * __temp[0]))
      * (1.0 / (tvV * __temp[1])))
   - ((tvCl2 * __temp[2])
      * (1.0 / (tvV * __temp[1]))));
  // w.r.t. A2
_jacobian[0 + _nrj*1] =
  -(((tvCl2 * __temp[2])
      *  -((1.0 / (tvV2 * __temp[3])))));
  // infusion rate for A1
_jacobian[0 + _nrj*2] = zzIRate[0];
// rate terms for A2
  // w.r.t. A1
_jacobian[1 + _nrj*0] =
   ((tvCl2 * __temp[2])
      * (1.0 / (tvV * __temp[1])));
  // w.r.t. A2
_jacobian[1 + _nrj*1] =
   ((tvCl2 * __temp[2])
      *  -((1.0 / (tvV2 * __temp[3]))));
  // infusion rate for A2
_jacobian[1 + _nrj*2] = zzIRate[1];
```

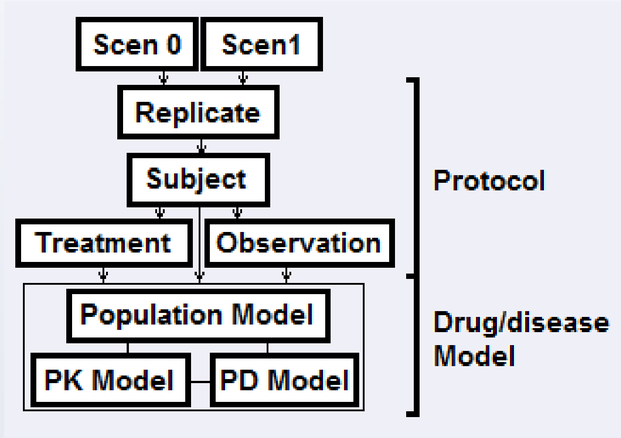
3. Embed models in R to facilitate scripted development

```
oneCpt <- ModelFragment({
  deriv( a <- - a * ke )
  dosepoint(a)
  c = a / v
})
mixMod01 <- ModelFragment({
  fixef(
      lke0 <- c(-6.9, -2.3, 0)
    , lv0 <- c(0, 2.3, 6.9)
    , ldVdWt <- c(-2.3, 0, 2.3)
    )
  ranef(diag(nlke, nlv) <- c(1,1))
  covariate(Wt)
  stparm(
      ke <- exp(lke0 + nlke)
    , v <- exp(lv0 + nlv) + Wt * exp(ldVdWt)
    )
})
errMod01 <- ModelFragment({
  error(eps0 <- 0.1)
  observe(cObs <- c*(1 + eps0))
})
# combine three fragments and create a model file
ModelGen("temp", myConcat(
    oneCpt, mixMod01, errMod01
    ))
```

4. Facilitate Trial Designer models by incorporating several fitted models, without error-prone transcription

```
 model  #### POPULATION MODEL
 popModel(){
 distrib(gender(male,female),wt,age
   = P( MVN((80(80-1,), 50),(10,0,11)) # male
      ,0.49
      , MVN((70(65,75), 50),(10,0,11)) # female
     )
   )
 }
```

(continued…)




```
model  #### PK MODEL
PK1(wt){
  dosepoint(aa)
  deriv(aa =  - aa * Ka)
  deriv(a1 = aa*Ka - a1*Ke)
  Ka = 5.2
  Cl = 20
  Ke = Cl / V
  covariate(wt)
  stparm(V = tvV * wt^1 * exp(nV))
  fixef(tvV = c(,1,))
  ranef(nV = .0324)
  c = a1 / V
  error(cEps = .01)
  observe(cObs = delay(c, 0) * exp(cEps))
}
model  #### PD MODEL
PD1(c){
  fixef(tvE0 = c(,0,)
    , tvEmax = c(,2.86,)
    , tvEc50 = c(,0.35,)
    , tvGamma = c(,1.46,)
    )
  stparm(e0 = tvE0)
  stparm(emax = tvEmax)
  stparm(ec50 = tvEc50)
  stparm(gamma = tvGamma)
  drugEffect = e0
       + (emax-e0) * (c^gamma / (c^gamma + ec50^gamma))
  ranef(nP = 1)
  deriv(T = 1, noss)
  placeboEffect = (1 - exp(- 0.3 * T)) * 0.3 * exp(-nP)
  totalEffect = drugEffect + placeboEffect
  p = 1 - exp(-totalEffect)
  x = ln(p/(1-p))
# alternative model
# x = ln(exp(max(totalEffect, 1e-30)) - 1)
  multi(PainRelief, ilogit, x)
}
################## DEFINE JOINT MODEL
join {
  pop:  popModel()
  pk1:  PK1( pop$wt )
  pd1:  PD1( pk1$c )
}
```

(continued…)

```
################## SUBJECT PROCEDURE
subject(tStop, dose){
  StartSubject()
  #### START TREATMENT & DOSE IN PARALLEL
  start treat(tStop, dose)
  start obs(tStop)
  sleep(tStop)
}
################## TREATMENT PROCEDURE
proc treat(tStop, dose){
  bolus(t, pk1$aa, dose)
}
################## OBSERVATION PROCEDURE
proc obs(tStop){
  sleep(0, 1); # <-- "1" make observations follow doses
  observe(t, pk1$cObs)
  observe(t, pd1$PainRelief);
  sleep(0.25)
  observe(t, pk1$cObs)
  observe(t, pd1$PainRelief);
  sleep(1.25)
  observe(t, pk1$cObs)
  observe(t, pd1$PainRelief);
}
################## DEFINE A GLOBAL VARIABLE
double (dDoseFactor)
################## REPLICATE PROCEDURE
proc replicate(tStop, nSub){
  InitReplicate()
  iArm = 0;  #### ARM 0
  for (iSub in 1:nSub){
    call subject(tStop, 0 * dDoseFactor);
  }
  iArm = 1;  #### ARM 1
  for (iSub in 1:nSub){
    call subject(tStop, 10 * dDoseFactor);
  }
}
################## TWO SCENARIO PROCEDURES
proc scen0(nRep, tStop, nSub){
  InitScenario()
  iScen = 0
  #### RUN A SERIES OF REPLICATES
  for (iRep in 1:nRep){
    call replicate(tStop, nSub)
  }
}
proc scen1(nRep, tStop, nSub){
  InitScenario()
  pd1$SCALEtvEc50 = 0.5
  iScen = 1
  for (iRep in 1:nRep){
    call replicate(tStop, nSub)
  }
}
################## MAIN PROCEDURE
proc main(){  #### RUN TWO SCENARIOS
  dDoseFactor = 1
  call scen0(100, 8, 40)
  call scen1(100, 8, 40)
}
```

References

Certara Corp.
Phoenix 1.4 Modeling Language Reference Guide.pdf