Modeling

May 21, 2012

Tim Bergsma

1 Purpose

This script runs NONMEM models and diagnostics for sample phase1 data.

2 Model Development

2.1 Set up for NONMEM run.

Listing 1:

```
> #Be sure to set directory to the script directory that contains this file.
> library(metrumrg)
> #command <- '/opt/NONMEM/nm72/nmqual/autolog.pl'
> cat.cov='SEX'
> cont.cov=c('HEIGHT','WEIGHT','AGE')
> par.list=c('CL','Q','KA','V','V2','V3')
> eta.list=paste('ETA',1:10,sep='')
```

2.2 Run NONMEM.

Listing 2:

```
> NONR72(
+       run=1001:1005,                      # 5 models, ctl pre-written
+       #command=command,                   # this version will search for NONMEM
+       project='../nonmem',                # must specify, unless ctl in getwd()
+       grid=TRUE,                          # set to FALSE for better error messaging (but slower)
+       nice=TRUE,                          # don't delete subversioned directories
+       checkrunno=FALSE,                   # TRUE auto-replaces conflicting run numbers
+       cont.cov=cont.cov,                  # see help for following
+       cat.cov=cat.cov,
```

```
+      par.list=par.list,
+      eta.list=eta.list,
+      grp='SEX',                            # separate diagnostic plots for each level of SEX
+      grpnames=c('female','male'),          # use these instead of 0, 1, when plotting by SEX
+      include.all=TRUE,                     # also show diagnostics with groups combined
+      plotfile='../nonmem/*/*.pdf',         # use the run dir and run name for the plot file
+      streams='../nonmem/ctl'               # expect the control streams here, not locally
+ )
```

```
Installing SIGCHLD signal handler...Done.
```

Listing 3:

```
> ready <- FALSE
> while (!ready){
+   state <- sapply(1001:1005,runstate,project='../nonmem')  # infer run progress from file presence/absence
+   print(state)
+   if(all(state=='done'))ready <- TRUE     #continue only when runs complete
+   Sys.sleep(3)                            #wait three seconds and check again
+ }
```

```
[1] "queued" "done"   "done"   "queued" "queued"
[1] "compiled" "compiled" "compiled" "compiled" "compiled"
[1] "compiled" "compiled" "compiled" "compiled" "compiled"
[1] "compiled" "compiled" "compiled" "compiled" "compiled"
[1] "compiled" "compiled" "compiled" "compiled" "compiled"
[1] "running" "running" "running" "running" "running"
[1] "running" "running" "running" "running" "running"
[1] "done"    "running" "running" "running" "running"
[1] "done"    "running" "running" "running" "running"
[1] "done"    "running" "running" "done"    "running"
[1] "done"    "running" "running" "done"    "done"
[1] "done"    "running" "running" "done"    "done"
[1] "done"    "running" "running" "done"    "done"
```

```
[1] "done"    "done"    "running" "done"    "done"
[1] "done" "done" "done" "done" "done"
```

Listing 4:

```
> Sys.sleep(10)                                #wait briefly to ensure all processes complete
```

Covariance succeeded on model 1005. We can make a quick run log using some simple tools. Table 1.

Listing 5:

```
> log <- rlog(1001:1005,'../nonmem',file=NULL) #don't want the 'wide' file, just the 'long' R object
> head(log)
```

```
  tool  run parameter   moment            value
1  nm7 1001       ofv  minimum 2526.39867230031
2  nm7 1001    THETA1 estimate          11.7167
3  nm7 1001    THETA1     prse             8.67
4  nm7 1001    THETA1       se          1.01636
5  nm7 1001    THETA2 estimate          14.5657
6  nm7 1001    THETA2     prse             8.67
```

Listing 6:

```
> tail(log)
```

```
    tool  run parameter   moment
245  nm7 1005  SIGMA2.2     prse
246  nm7 1005  SIGMA2.2       se
247  nm7 1005       cov   status
248  nm7 1005      prob     text
249  nm7 1005       min   status
250  nm7 1005      data filename
                                                value
245                                              33.5
```

```
246                                                0.0676412
247                                                        0
248 1005 phase1 2 CMT like 1004 but diff. initial on V3
249                                                        0
250                          ../../data/derived/phase1.csv
```

Listing 7:

```
> sapply(log,class)
```

```
       tool         run   parameter      moment       value
"character"   "integer" "character" "character" "character"
```

Listing 8:

```
> log$tool <- NULL
> unique(log$parameter)
```

```
 [1] "ofv"      "THETA1"   "THETA2"   "THETA3"   "OMEGA1.1" "OMEGA2.1"
 [7] "OMEGA2.2" "OMEGA3.1" "OMEGA3.2" "OMEGA3.3" "SIGMA1.1" "SIGMA2.1"
[13] "SIGMA2.2" "cov"      "prob"     "min"      "data"     "THETA4"
[19] "THETA5"   "OMEGA4.1" "OMEGA4.2" "OMEGA4.3" "OMEGA4.4" "OMEGA5.1"
[25] "OMEGA5.2" "OMEGA5.3" "OMEGA5.4" "OMEGA5.5" "THETA6"   "THETA7"
```

Listing 9:

```
> log <- log[log$parameter %in% c('ofv','prob','cov','min'),]
> log
```

```
     run parameter  moment
1   1001       ofv minimum
38  1001       cov  status
39  1001      prob    text
40  1001       min  status
42  1002       ofv minimum
```

```
112 1002       cov  status
113 1002      prob    text
114 1002       min  status
116 1003       ofv minimum
153 1003       cov  status
154 1003      prob    text
155 1003       min  status
157 1004       ofv minimum
194 1004       cov  status
195 1004      prob    text
196 1004       min  status
198 1005       ofv minimum
247 1005       cov  status
248 1005      prob    text
249 1005       min  status
```

	value
1	2526.39867230031
38	0
39	1001 phase1 1CMT
40	0
42	2525.96526753388
112	1
113	1002 phase1 2 CMT
114	1
116	2569.89393760215
153	1
154	1003 phase1 2 CMT like 1002 but no eta on Q/v3 and no + err
155	0
157	2570.45022637547
194	0
195	1004 phase1 2 CMT like 1003 but better bounds
196	0
198	2405.91625845151
247	0

```
248         1005 phase1 2 CMT like 1004 but diff. initial on V3
249                                                          0
```

Listing 10:

```
> with(log, constant(moment,within=parameter))#i.e., moment is non-informative here.

[1] TRUE
```

Listing 11:

```
> log <- data.frame(cast(log,run ~ parameter))
> log <- shuffle(log,'prob','run')
> log$ofv <- signif(as.numeric(as.character(log$ofv,6)))
```

Table 1: Run Log

run	prob	cov	min	ofv
1001	1001 phase1 1CMT	0	0	2526.40
1002	1002 phase1 2 CMT	1	1	2525.97
1003	1003 phase1 2 CMT like 1002 but no eta on Q/v3 and no + err	1	0	2569.89
1004	1004 phase1 2 CMT like 1003 but better bounds	0	0	2570.45
1005	1005 phase1 2 CMT like 1004 but diff. initial on V3	0	0	2405.92

3 Predictive Check

3.1 Create a simulation control stream.

Convert control stream to R object.

Listing 12:

```
> ctl <- read.nmctl('../nonmem/ctl/1005.ctl')
```

Strip comments and view.

Listing 13:

```
> ctl[] <- lapply(ctl,function(rec)sub(' *;.*','',rec))          # read control stream into a list
> ctl                                                            # print it like text
```

```
 [1] "$PROB 1005 phase1 2 CMT like 1004 but diff. initial on V3"
 [2] "$INPUT C ID TIME SEQ=DROP EVID AMT DV SUBJ HOUR TAFD TAD LDOS MDV HEIGHT WT SEX AGE DOSE FED"
 [3] "$DATA ../../data/derived/phase1.csv IGNORE=C"
 [4] "$SUBROUTINE ADVAN4 TRANS4"
 [5] "$PK"
 [6] " CL=THETA(1)*EXP(ETA(1)) * THETA(6)**SEX * (WT/70)**THETA(7)"
 [7] " V2 =THETA(2)*EXP(ETA(2))"
 [8] " KA=THETA(3)*EXP(ETA(3))"
 [9] " Q  =THETA(4)"
[10] " V3=THETA(5)"
[11] " S2=V2"
[12] " "
[13] "$ERROR"
[14] " Y=F*(1+ERR(1)) + ERR(2)"
[15] " IPRE=F"
[16] ""
[17] "$THETA"
[18] "(0,10,50)"
[19] "(0,10,100)"
[20] "(0,0.2, 5)"
[21] "(0,10,50)"
[22] "(0,100,1000)"
[23] "(0,1,2)"
[24] "(0,0.75,3)"
[25] ""
```

```
[26] "$OMEGA BLOCK(3)"
[27] ".1"
[28] ".01 .1"
[29] ".01 .01 .1"
[30] ""
[31] ""
[32] ""
[33] ""
[34] ""
[35] ""
[36] ""
[37] ""
[38] "$SIGMA 0.1 0.1"
[39] ""
[40] ""
[41] ""
[42] ""
[43] "$ESTIMATION MAXEVAL=9999 PRINT=5 NOABORT METHOD=1 INTER MSFO=./1005.msf"
[44] "$COV PRINT=E"
[45] "$TABLE NOPRINT FILE=./1005.tab ONEHEADER ID AMT TIME EVID PRED IPRE CWRES"
[46] "$TABLE NOPRINT FILE=./1005par.tab ONEHEADER ID TIME CL Q V2 V3 KA ETA1 ETA2 ETA3"
[47] ""
[48] ""
[49] ""
[50] ""
[51] ""
[52] ""
[53] ""
[54] ""
[55] ""
[56] ""
[57] ""
[58] ""
[59] ""
```

```
[60] ""
[61] ""
[62] ""
[63] ""
```

Fix records of interest.

Listing 14:

```
> ctl$prob                                                    # problem statement

[1] "1005 phase1 2 CMT like 1004 but diff. initial on V3"
```

Listing 15:

```
> ctl$prob <- sub('1005','1105',ctl$prob)                     # substitute new run number
> names(ctl)

 [1] "prob"       "input"      "data"       "subroutine" "pk"
 [6] "error"      "theta"      "omega"      "sigma"      "estimation"
[11] "cov"        "table"      "table"
```

Listing 16:

```
> names(ctl)[names(ctl)=='theta'] <- 'msfi'                   # replace theta with final msfi
> ctl$msfi <- '=../1005/1005.msf'
> ctl$omega <- NULL                                           # drop omega, sigma
> ctl$sigma <- NULL
> names(ctl)[names(ctl)=='estimation'] <- 'simulation'        # simulate instead of estimate
> ctl$simulation <- 'ONLYSIM (1968) SUBPROBLEMS=500'
> ctl$cov <- NULL                                            # drop covariance step
> ctl$table <- NULL                                          # replace multiple tables with one
> ctl$table <- NULL
> ctl$table <- 'DV NOHEADER NOPRINT FILE=./1105.tab FORWARD NOAPPEND' # only really need DV, save file space
> write.nmctl(ctl,'../nonmem/ctl/1105.ctl')
```

3.2 Run the simulation.

This run makes the predictions (simulations).

Listing 17:

```
> NONR72(
+      run=1105,
+      #command=command,
+      project='../nonmem',
+      grid=TRUE,
+      nice=TRUE,
+      diag=FALSE,
+      streams='../nonmem/ctl'
+ )
> while (runstate(1105,project='../nonmem')!='done')Sys.sleep(5)
> Sys.sleep(30) # let all processes complete
```

3.3 Combine the original data and the simulation data.

Now we fetch the results and integrate them with the other data.

Listing 18:

```
> x <- superset(
+   run=1105,
+   project='../nonmem',
+   read.output=list(read.table,header=FALSE)
+ )
> x <- x[,c('SUBJ','TIME','DV','V1','1105')]
> read.nmctl('../nonmem/1105/1105.ctl')$simulation

[1] "ONLYSIM (1968) SUBPROBLEMS=500"
```

Listing 19:

```
> x$SIM <- rep(1:500,each=nrow(x)/500)
> colname(x) <- c(V1='PRED')
> x <- x[x$`1105`==1,]
> x$`1105` <- NULL
> head(x)
```

```
  SUBJ TIME    DV    PRED SIM
2    1 0.00     . 0.00000   1
3    1 0.25 0.363 0.72542   1
4    1 0.50 0.914 1.38320   1
5    1 1.00  1.12 2.06720   1
6    1 2.00  2.28 3.48570   1
7    1 3.00  1.63 5.44600   1
```

Listing 20:

```
> nrow(x)
```

```
[1] 275000
```

Listing 21:

```
> str(x)
```

```
'data.frame':   275000 obs. of  5 variables:
 $ SUBJ: int  1 1 1 1 1 1 1 1 1 1 ...
 $ TIME: num  0 0.25 0.5 1 2 3 4 6 8 12 ...
 $ DV  : chr  "." "0.363" "0.914" "1.12" ...
 $ PRED: num  0 0.725 1.383 2.067 3.486 ...
 $ SIM : int  1 1 1 1 1 1 1 1 1 1 ...
```

Listing 22:

```
> x <- x[x$DV != '.',]
> x$DV <- as.numeric(x$DV)
```

3.4 Plot predictive checks.

3.4.1 Aggregate data within subject.

Since subjects may contribute differing numbers of observations, it may be useful to look at predictions from a subject-centric perspective. Therefore, we wish to calculate summary statistics for each subject, (observed and predicted) and then make obspred comparisons therewith.

Listing 23:

```
> head(x)
```

```
  SUBJ TIME    DV    PRED SIM
3    1 0.25 0.363 0.72542   1
4    1 0.50 0.914 1.38320   1
5    1 1.00 1.120 2.06720   1
6    1 2.00 2.280 3.48570   1
7    1 3.00 1.630 5.44600   1
8    1 4.00 2.040 2.99140   1
```

Listing 24:

```
> subject <- melt(x,measure.var=c('DV','PRED'))
> head(subject)
```

```
  SUBJ TIME SIM variable value
1    1 0.25   1       DV 0.363
2    1 0.50   1       DV 0.914
3    1 1.00   1       DV 1.120
4    1 2.00   1       DV 2.280
5    1 3.00   1       DV 1.630
6    1 4.00   1       DV 2.040
```

We are going to aggregate each subject's DV and PRED values using cast(). cast() likes an aggregation function that returns a list. We write one that grabs min med max for each subject, sim, and variable.

Listing 25:

```
> metrics <- function(x)list(min=min(x), med=median(x), max=max(x))
```

Now we cast, ignoring time.

Listing 26:

```
> subject <- data.frame(cast(subject, SUBJ + SIM + variable  ~  .,fun=metrics))
> head(subject)
```

```
  SUBJ SIM variable       min    med     max
1    1   1       DV  0.363000 1.6100  3.0900
2    1   1     PRED  0.725420 3.4795  5.4460
3    1   2       DV  0.363000 1.6100  3.0900
4    1   2     PRED -0.085238 2.2941  4.6468
5    1   3       DV  0.363000 1.6100  3.0900
6    1   3     PRED -0.022407 4.8896 12.3770
```

Note that regardless of SIM, DV (observed) is constant.

Now we melt the metrics.

Listing 27:

```
> metr <- melt(subject,measure.var=c('min','med','max'),variable_name='metric')
> head(metr)
```

```
  SUBJ SIM variable metric     value
1    1   1       DV    min  0.363000
2    1   1     PRED    min  0.725420
3    1   2       DV    min  0.363000
4    1   2     PRED    min -0.085238
5    1   3       DV    min  0.363000
6    1   3     PRED    min -0.022407
```

Listing 28:

```
> metr$value <- reapply(
+       metr$value,
+       INDEX=metr[,c('SIM','variable','metric')],
+       FUN=sort,
+       na.last=FALSE
+ )
> metr <- data.frame(cast(metr))
> head(metr)
```

```
  SUBJ SIM metric    DV     PRED
1    1   1    min 0.139 -0.61537
2    1   1    med 1.025  1.25865
3    1   1    max 2.530  2.17620
4    1   2    min 0.139 -0.35196
5    1   2    med 1.025  1.20926
6    1   2    max 2.530  2.42390
```

Listing 29:

```
> nrow(metr)
```

```
[1] 60000
```

Listing 30:

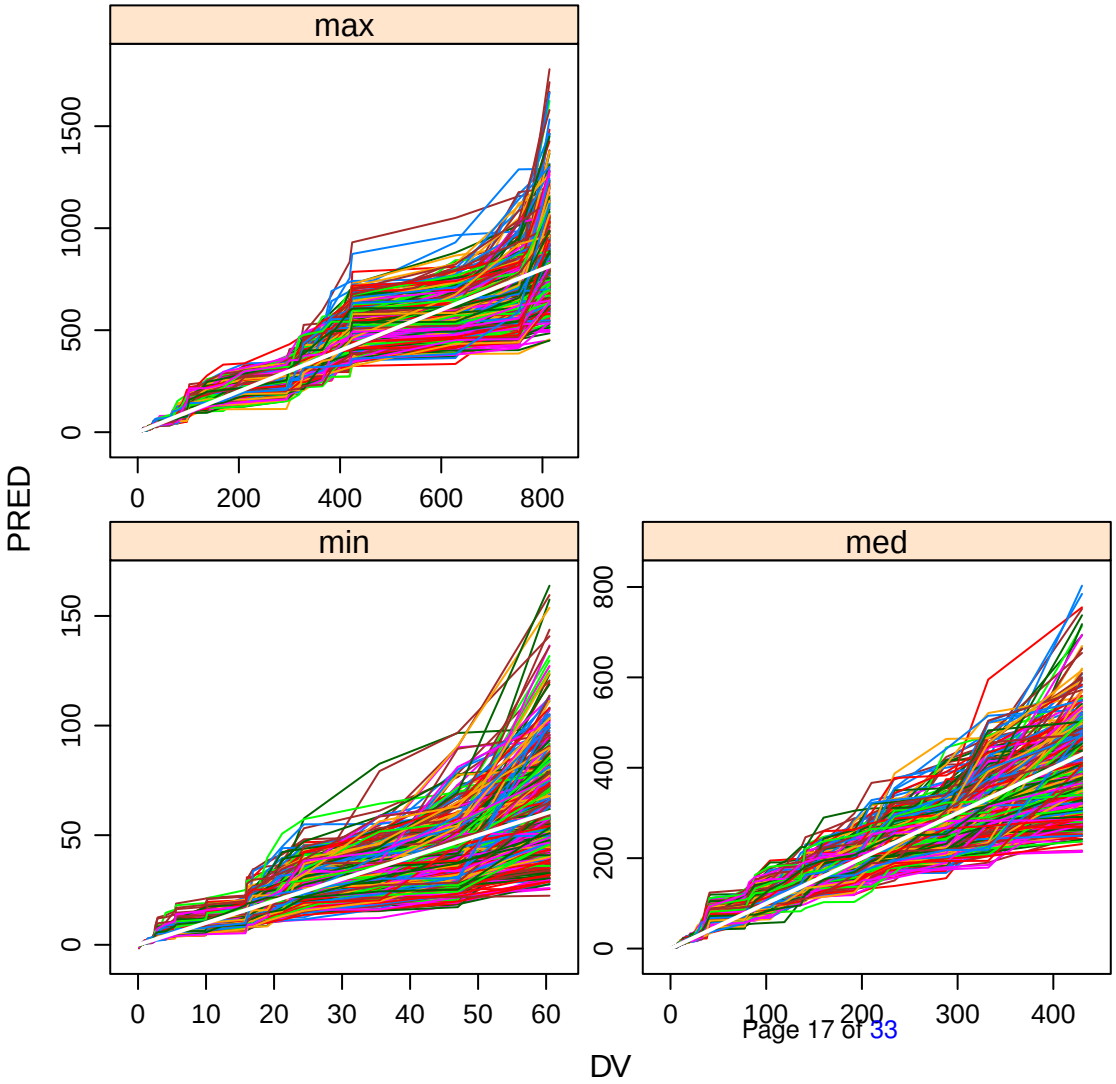
```
> metr <- metr[!is.na(metr$DV),]#maybe no NA
> nrow(metr)
```

```
[1] 60000
```

We plot using lattice.

Listing 31:

```
> print(
+       xyplot(
+               PRED ~ DV|metric,
+               metr,
+               groups=SIM,
+               scales=list(relation='free'),
+               type='l',
+               panel=function(...){
+                       panel.superpose(...)
+                       panel.abline(0,1,col='white',lwd=2)
+               }
+       )
+ )
```


max
min
med
PRED
DV
0
500
1000
1500
0
200
400
600
800
0
50
100
150
0
10
20
30
40
50
60
0
200
400
600
800
0
100
200
300
400

For detail, we show one endpoint, tossing the outer 5 percent of values, and indicating quartiles.

Listing 32:

```
> med <- metr[metr$metric=='med',]
> med$metric <- NULL
> head(med)
```

```
   SUBJ SIM    DV    PRED
2     1   1 1.025 1.25865
5     1   2 1.025 1.20926
8     1   3 1.025 1.57990
11    1   4 1.025 0.88489
14    1   5 1.025 1.65875
17    1   6 1.025 0.95005
```

Listing 33:

```
> trim <- inner(med, id.var=c('SIM'),measure.var=c('PRED','DV'))
> head(trim)
```

```
  SIM DV PRED
1   1 NA   NA
2   2 NA   NA
3   3 NA   NA
4   4 NA   NA
5   5 NA   NA
6   6 NA   NA
```

Listing 34:

```
> nrow(trim)
```

```
[1] 20000
```

Listing 35:

```
> trim <- trim[!is.na(trim$DV),]
> nrow(trim)

[1] 19000
```

Listing 36:

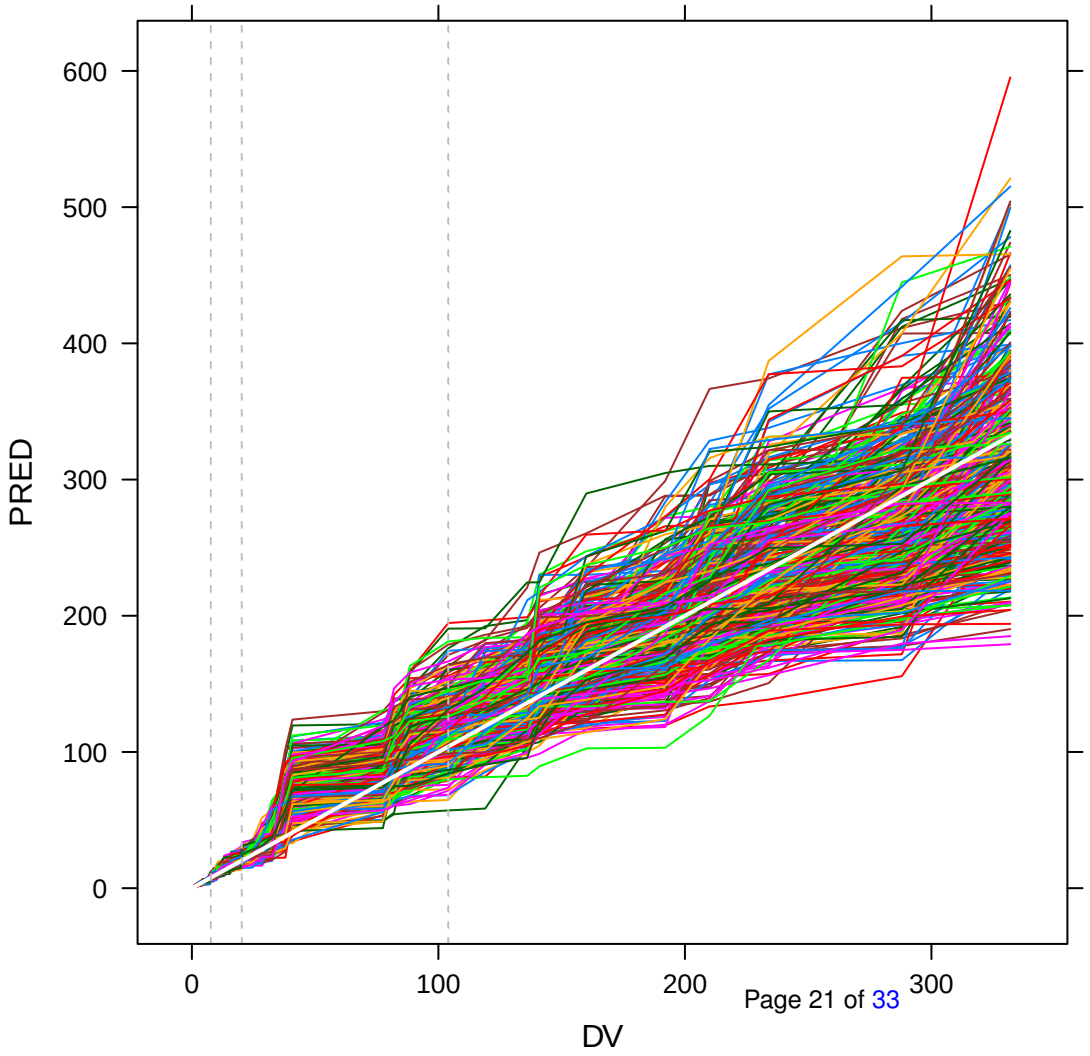
```
> head(trim)

    SIM   DV    PRED
501   1 1.13 2.05880
502   2 1.13 2.00535
503   3 1.13 1.65480
504   4 1.13 1.06910
505   5 1.13 2.05960
506   6 1.13 0.98589
```

Listing 37:

```
> print(
+       xyplot(
+               PRED ~ DV,
+               trim,
+               groups=SIM,
+               type='l',
+               panel=function(x,y,...){
+                       panel.xyplot(x=x,y=y,...)
+                       panel.abline(0,1,col='white',lwd=2)
+                       panel.abline(
+                               v=quantile(x,probs=c(0.25,0.5,0.75)),
+                               col='grey',
+                               lty=2
+                       )
```

```
+                     }
+           )
+ )
```


PRED
DV
0
100
200
300
400
500
600
0
100
200
300

We also show densityplots of predictions at those quartiles.

Listing 38:

```
> head(trim)

    SIM   DV    PRED
501   1 1.13 2.05880
502   2 1.13 2.00535
503   3 1.13 1.65480
504   4 1.13 1.06910
505   5 1.13 2.05960
506   6 1.13 0.98589
```

Listing 39:

```
> quantile(trim$DV)

    0%    25%    50%    75%   100%
  1.13   7.69  20.25 104.00 332.00
```

Listing 40:

```
> molt <- melt(trim, id.var='SIM')
> head(molt)

  SIM variable value
1   1       DV  1.13
2   2       DV  1.13
3   3       DV  1.13
4   4       DV  1.13
5   5       DV  1.13
6   6       DV  1.13
```

Listing 41:

```
> quart <- data.frame(cast(molt,SIM+variable ~ .,fun=quantile,probs=c(0.25,0.5,0.75)))
> head(quart)
```

```
  SIM variable     X25.     X50.      X75.
1   1       DV  7.95000 20.25000 100.10000
2   1     PRED 11.92825 22.16750 103.96500
3   2       DV  7.95000 20.25000 100.10000
4   2     PRED  7.23495 20.27050 105.20875
5   3       DV  7.95000 20.25000 100.10000
6   3     PRED  7.82690 14.50425  98.27575
```

Listing 42:

```
> molt <- melt(quart,id.var='variable',measure.var=c('X25.','X50.','X75.'),variable_name='quartile')
> head(molt)
```

```
  variable quartile    value
1       DV     X25.  7.95000
2     PRED     X25. 11.92825
3       DV     X25.  7.95000
4     PRED     X25.  7.23495
5       DV     X25.  7.95000
6     PRED     X25.  7.82690
```

Listing 43:

```
> levels(molt$quartile)
```

```
[1] "X25." "X50." "X75."
```

Listing 44:

```
> levels(molt$quartile) <- c('first quartile','second quartile','third quartile')
> head(molt)
```

```
  variable       quartile    value
1       DV first quartile  7.95000
2     PRED first quartile 11.92825
3       DV first quartile  7.95000
4     PRED first quartile  7.23495
5       DV first quartile  7.95000
6     PRED first quartile  7.82690
```

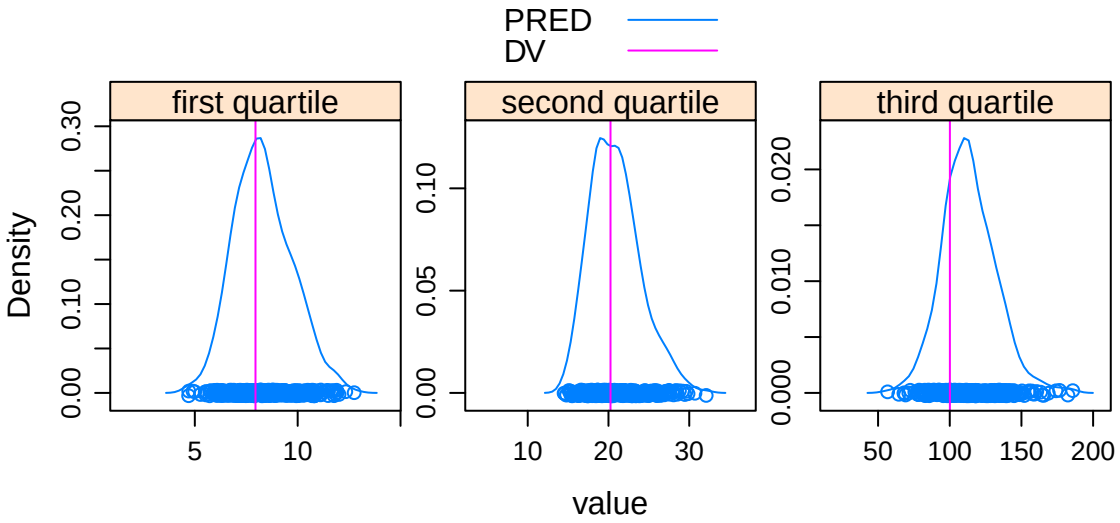
Listing 45:

```
> levels(molt$variable)

[1] "DV"   "PRED"
```

Listing 46:

```
> molt$variable <- factor(molt$variable,levels=c('PRED','DV'))
> print(
+       densityplot(
+                ~value|quartile,
+               molt,
+               groups=variable,
+               layout=c(3,1),
+               scales=list(relation='free'),
+               aspect=1,
+               panel=panel.superpose,
+               panel.groups=function(x,...,group.number){
+                       if(group.number==1)panel.densityplot(x,...)
+                       if(group.number==2)panel.abline(v=unique(x),...)
+               },
+               auto.key=TRUE
+       )
+ )
```


PRED
DV
first quartile
second quartile
third quartile
Density
0.00
0.10
0.20
0.30
0.05
0.020
0.010
0.000
5
10
20
30
50
100
150
200
value

4 Bootstrap Estimates of Parameter Uncertainty

4.1 Create directories.

Listing 47:

```
> getwd()
```

```
[1] "/data/project/metrumrg/inst/example/project/script"
```

Listing 48:

```
> dir.create('../nonmem/1005boot')
> dir.create('../nonmem/1005bootdata')
> dir.create('../nonmem/1005bootctl')
```

4.2 Create replicate control streams.

Listing 49:

```
> ctl <- clear(readLines('../nonmem/ctl/1005.ctl'),';.+',fixed=FALSE)
> #ctl <- read.nmctl('../nonmem/1005/1005.ctl')
> ctl <- as.nmctl(ctl)
> names(ctl)
```

```
 [1] "prob"       "input"      "data"       "subroutine" "pk"
 [6] "error"      "theta"      "omega"      "sigma"      "estimation"
[11] "cov"        "table"      "table"
```

Listing 50:

```
> ctl$cov <- NULL
> ctl$table <- NULL
> ctl$table <- NULL
> ctl$prob
```

```
[1] "1005 phase1 2 CMT like 1004 but diff. initial on V3"
```

Listing 51:

```
> ctl$data
```

```
[1] "../../data/derived/phase1.csv IGNORE=C"
```

Listing 52:

```
> run <- sprintf('%03.0f',1:300)
> invisible(
+   lapply(
+     run,
+     function(i,ctl){
+       ctl$prob <- sub('1005',i,ctl$prob)
+       ctl$data <- sub(
+         '../../data/derived/phase1.csv',
+         sub('\\*',i,'../../1005bootdata/*.csv'),
+         ctl$data
+       )
+       write.nmctl(ctl,file=glue('../nonmem/1005bootctl/',i,'.ctl'))
+     },
+     ctl=ctl
+   )
+ )
```

4.3 Create replicate data sets by resampling original.

Listing 53:

```
>  bootset <- read.csv('../data/derived/phase1.csv')
>  r <- resample(
+       bootset,
+       names=run,
```

```
+       key='ID',
+       rekey=TRUE,
+       out='../nonmem/1005bootdata',
+       stratify='SEX'
+  )
```

4.4 Run bootstrap models.

Listing 54:

```
> NONR72(
+      run=run,
+      boot=TRUE,
+      project='../nonmem/1005boot',
+      streams='../nonmem/1005bootctl'
+ )
```

```
1] "Run 285 has exit code 0"
 "Run 116 has exit code 0"
```

Listing 55:

```
> length(dir('../nonmem/1005boot'))
```

```
[1] 300
```

Listing 56:

```
> system('qstat -f')
> ready <- FALSE
> while (!ready){
+   state <- sapply(run,runstate,project='../nonmem/1005boot')  # infer run progress from file presence/absence
+   print(table(state))
+   if(all(state=='done'))ready <- TRUE     #continue only when runs complete
+   Sys.sleep(10)                            #wait 10 seconds and check again
+ }
```

```
state
compiled   queued
      40      260
state
compiled   queued  running
      32      260        8
state
compiled     done   queued  running
      21        7      260       12
state
compiled     done   queued  running
      12       21      253       14
state
compiled     done   queued  running
      20       38      240        2
state
compiled     done   queued  running
      34       40      225        1
state
compiled     done   queued  running
      31       40      221        8
state
     compiled          done indeterminate        queued       running
           22            48             1           218            11
state
compiled     done   queued  running
      22       58      212        8
state
compiled     done   queued  running
      15       71      207        7
state
compiled     done   queued  running
      11       79      199       11
state
```

```
compiled     done   queued  running
      18       90      189        3
state
compiled     done   queued  running
      28       92      176        4
state
compiled     done   queued  running
      27       95      172        6
state
     compiled          done indeterminate        queued       running
            6           102             1           171            20
state
compiled     done   queued  running
       9      122      164        5
state
compiled     done   queued  running
      27      127      143        3
state
compiled     done   queued  running
      25      130      138        7
state
compiled     done   queued  running
      17      136      134       13
state
compiled     done   queued  running
      11      148      130       11
state
compiled     done   queued  running
      15      161      120        4
state
compiled     done   queued  running
      20      165      108        7
state
compiled     done   queued  running
```

```
      22      172       99        7
state
compiled     done   queued  running
      15      181       95        9
state
compiled     done   queued  running
      16      188       85       11
state
compiled     done   queued  running
      14      198       81        7
state
compiled     done   queued  running
      16      205       69       10
state
compiled     done   queued  running
      18      215       61        6
state
compiled     done   queued  running
      20      221       54        5
state
compiled     done   queued  running
      22      227       45        6
state
compiled     done   queued  running
      14      235       38       13
state
compiled     done   queued  running
      12      246       31       11
state
compiled     done   queued  running
      21      257       18        4
state
compiled     done   queued  running
      21      260       11        8
```

```
state
compiled     done   queued  running
      15      266        4       15
state
compiled     done  running
      10      281        9
state
compiled     done  running
       3      292        5
state
   done running
    299       1
state
done
 300
```

Listing 57:

```
> boot <- rlog(
+       run=run,
+       project='../nonmem/1005boot',
+       boot=TRUE,
+       append=FALSE,
+       tool='nm7',
+   file=NULL
+ )
> write.csv(boot, '../nonmem/1005bootlog.csv')
```

5 File Disposition

Predictive checks and bootstraps make huge files that need not be retained.

Listing 58:

```
> unlink('../nonmem/1105',recursive=TRUE)
```

```
> unlink('../nonmem/1005boot',recursive=TRUE)
> unlink('../nonmem/1005bootdata',recursive=TRUE)
> unlink('../nonmem/1005bootctl',recursive=TRUE)
```