MIfuns Sample Script

Phase I Modeling

October 20, 2010

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:

```
> getwd()
```

```
[1] "/Users/timb/project/metrum-mifuns/inst/sample/script"
```

Listing 2:

```
> library(MIfuns)
```

```
MIfuns 4.1.0
```

Listing 3:

```
> command <- '/common/NONMEM/nm7_osx1/test/nm7_osx1.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.

To force a re-run of this model, delete 1005/diagnostics.pdf.

Listing 4:

```
> if(!file.exists('../nonmem/1005/diagnostics.pdf'))NONR(
+      run=1005,
+      command=command,
+      project='../nonmem',
+      grid=TRUE,
+      nice=TRUE,
+      checkrunno=FALSE,
+      cont.cov=cont.cov,
+      cat.cov=cat.cov,
+      par.list=par.list,
+      eta.list=eta.list,
+      plotfile='../nonmem/*/diagnostics.pdf',
+      streams='../nonmem/ctl'
+ )
> getwd()
```

```
[1] "/Users/timb/project/metrum-mifuns/inst/sample/script"
```

Listing 5:

```
> while(!file.exists('../nonmem/1005/diagnostics.pdf')){}
```

Covariance succeeded on model 1005.

3 Predictive Check

3.1 Create a simulation control stream.

Convert control stream to R object.

Listing 6:

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

Strip comments and view.

Listing 7:

```
> ctl[] <- lapply(ctl,function(rec)sub(' *;.*','',rec))
> ctl
```

```
 [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] "$PK  CL=THETA(1)*EXP(ETA(1)) * THETA(6)**SEX * (WT/70)**THETA(7)"
 [6] " V2 =THETA(2)*EXP(ETA(2))"
 [7] " KA=THETA(3)*EXP(ETA(3))"
 [8] " Q  =THETA(4)"
 [9] " V3=THETA(5)"
[10] " S2=V2"
[11] " "
[12] "$ERROR  Y=F*EXP(ERR(1))"
[13] " IPRE=F"
[14] ""
[15] "$THETA (0,10,50)"
[16] "(0,10,100)"
[17] "(0,0.2, 5)"
[18] "(0,10,50)"
[19] "(0,100,1000)"
[20] "(0,1,2)"
[21] "(0,0.75,3)"
[22] ""
[23] "$OMEGA 0.09 0.09 0.09 "
```

```
[24] ""
[25] ""
[26] ""
[27] ""
[28] ""
[29] "$SIGMA 0.09"
[30] ""
[31] ""
[32] ""
[33] "$ESTIMATION MAXEVAL=9999 PRINT=5 NOABORT METHOD=1 INTER MSFO=./1005.msf"
[34] "$COV PRINT=E"
[35] "$TABLE NOPRINT FILE=./1005.tab ONEHEADER ID AMT TIME EVID PRED IPRE CWRES"
[36] "$TABLE NOPRINT FILE=./1005par.tab ONEHEADER ID TIME CL Q V2 V3 KA ETA1 ETA2
   ETA3"
```

Fix records of interest.

Listing 8:

```
> ctl$prob

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

Listing 9:

```
> ctl$prob <- sub('1005','1105',ctl$prob)
> names(ctl)

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

Listing 10:

```
> names(ctl)[names(ctl)=='theta'] <- 'msfi'
> ctl$msfi <- '=../1005/1005/msf'
> ctl$omega <- NULL
> ctl$sigma <- NULL
> names(ctl)[names(ctl)=='estimation'] <- 'simulation'
> ctl$simulation <- 'ONLYSIM (1968) SUBPROBLEMS=500'
> ctl$cov <- NULL
> ctl$table <- NULL
> ctl$table <- NULL
> ctl$table <- 'DV NOHEADER NOPRINT FILE=./1105.tab FORWARD NOAPPEND'
> write.nmcontrol('../nonmem/ctl/1105.ctl')
```

3.2 Run the simulation.

This run makes the predictions (simulations).

Listing 11:

```
> if(!file.exists('../nonmem/1105/1105.lst'))NONR(
+      run=1105,
+      command=command,
+      project='../nonmem',
+      grid=TRUE,
+      nice=TRUE,
+      diag=FALSE,
+      streams='../nonmem/ctl'
+ )
> getwd()
```

```
[1] "/Users/timb/project/metrum-mifuns/inst/sample/script"
```

Listing 12:

```
> while(!file.exists('../nonmem/1105/1105.lst')){}
```

3.3 Recover and format the original dataset.

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

Listing 13:

```
> phase1 <- read.csv('../data/derived/phase1.csv',na.strings='.')
> head(phase1)
```

```
     C ID TIME SEQ EVID  AMT    DV SUBJ HOUR TAFD  TAD LDOS MDV HEIGHT WEIGHT
1    C  1 0.00   0    0   NA 0.000    1 0.00 0.00   NA   NA   0    174   74.2
2 <NA>  1 0.00   1    1 1000    NA    1 0.00 0.00 0.00 1000   1    174   74.2
3 <NA>  1 0.25   0    0   NA 0.363    1 0.25 0.25 0.25 1000   0    174   74.2
4 <NA>  1 0.50   0    0   NA 0.914    1 0.50 0.50 0.50 1000   0    174   74.2
5 <NA>  1 1.00   0    0   NA 1.120    1 1.00 1.00 1.00 1000   0    174   74.2
6 <NA>  1 2.00   0    0   NA 2.280    1 2.00 2.00 2.00 1000   0    174   74.2
  SEX  AGE DOSE FED SMK DS CRCN predose zerodv
1   0 29.1 1000   1   0  0 83.5       1      1
2   0 29.1 1000   1   0  0 83.5       0      0
3   0 29.1 1000   1   0  0 83.5       0      0
4   0 29.1 1000   1   0  0 83.5       0      0
5   0 29.1 1000   1   0  0 83.5       0      0
6   0 29.1 1000   1   0  0 83.5       0      0
```

Listing 14:

```
> phase1 <- phase1[is.na(phase1$C),c('SUBJ','TIME','DV')]
> records <- nrow(phase1)
> records
```

```
[1] 550
```

Listing 15:

```
> phase1 <- phase1[rep(1:records,500),]
> nrow(phase1)

[1] 275000
```

Listing 16:

```
> phase1$SIM <- rep(1:500,each=records)
> #head(phase1,300)
> with(phase1,DV[SIM==1 & SUBJ==12])

 [1]     NA  2.260  2.830  8.730 19.300 15.200 16.200  8.830 12.900 12.700
[11]  7.140  5.740  1.980  0.791
```

Listing 17:

```
> with(phase1,DV[SIM==2 & SUBJ==12])

 [1]     NA  2.260  2.830  8.730 19.300 15.200 16.200  8.830 12.900 12.700
[11]  7.140  5.740  1.980  0.791
```

3.4 Recover and format the simulation results.

Listing 18:

```
> pred <- scan('../nonmem/1105/1105.tab')
> nrow(phase1)

[1] 275000
```

Listing 19:

```
> length(pred)

[1] 275000
```

3.5 Combine the original data and the simulation data.

Listing 20:

```
> phase1$PRED <- pred
> head(phase1)

  SUBJ TIME    DV SIM    PRED
2    1 0.00    NA   1 0.00000
3    1 0.25 0.363   1 0.17932
4    1 0.50 0.914   1 0.53642
5    1 1.00 1.120   1 0.78983
6    1 2.00 2.280   1 1.84990
7    1 3.00 1.630   1 1.96530
```

Listing 21:

```
> phase1 <- phase1[!is.na(phase1$DV),]
> head(phase1)
```

```
  SUBJ TIME    DV SIM    PRED
3    1 0.25 0.363   1 0.17932
4    1 0.50 0.914   1 0.53642
5    1 1.00 1.120   1 0.78983
6    1 2.00 2.280   1 1.84990
7    1 3.00 1.630   1 1.96530
8    1 4.00 2.040   1 2.01810
```

3.6 Plot predictive checks.

3.6.1 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 22:

```
> head(phase1)
```

```
  SUBJ TIME    DV SIM    PRED
3    1 0.25 0.363   1 0.17932
4    1 0.50 0.914   1 0.53642
5    1 1.00 1.120   1 0.78983
6    1 2.00 2.280   1 1.84990
7    1 3.00 1.630   1 1.96530
8    1 4.00 2.040   1 2.01810
```

Listing 23:

```
> subject <- melt(phase1,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 24:

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

Now we cast, ignoring time.

Listing 25:

```
> 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 0.179320 1.9653 5.0314
3    1   2       DV 0.363000 1.6100 3.0900
4    1   2     PRED 0.096462 3.0448 7.4728
5    1   3       DV 0.363000 1.6100 3.0900
6    1   3     PRED 0.450430 5.5284 8.7665
```

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

Now we melt the metrics.

Listing 26:

```
> 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 0.179320
3    1   2       DV    min 0.363000
4    1   2     PRED    min 0.096462
5    1   3       DV    min 0.363000
6    1   3     PRED    min 0.450430
```

Listing 27:

```
> 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 0.064213
2    1   1    med 1.025 1.943600
3    1   1    max 2.530 3.945400
4    1   2    min 0.139 0.016162
5    1   2    med 1.025 1.476300
6    1   2    max 2.530 3.463200
```

Listing 28:

```
> nrow(metr)
```

```
[1] 60000
```

Listing 29:

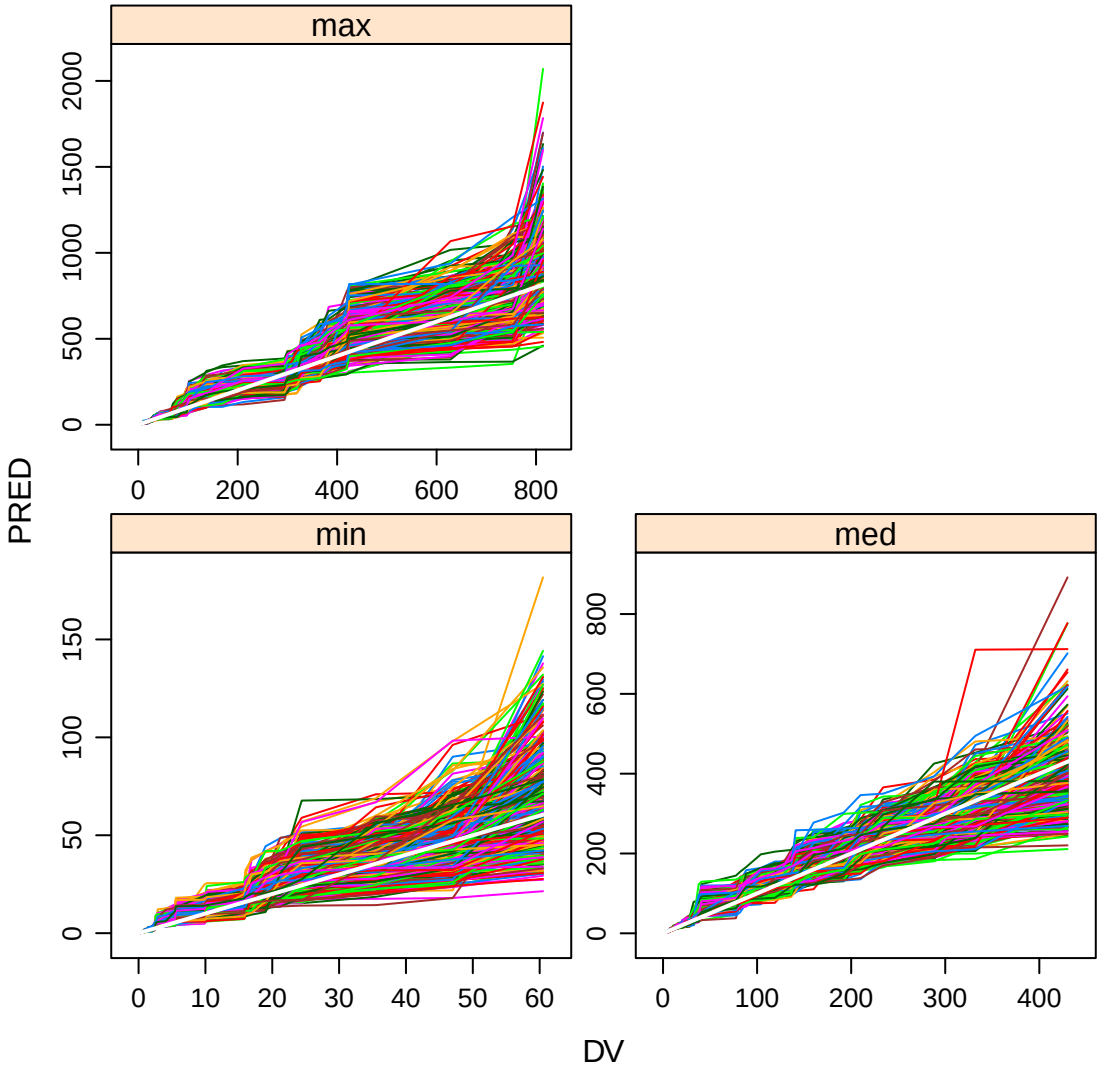
```
> metr <- metr[!is.na(metr$DV),]#maybe no NA
> nrow(metr)
```

```
[1] 60000
```

We plot using lattice.

Listing 30:

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



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

Listing 31:

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

```
   SUBJ SIM    DV     PRED
2     1   1 1.025 1.943600
5     1   2 1.025 1.476300
```

```
8     1   3 1.025 1.466300
11    1   4 1.025 1.342400
14    1   5 1.025 1.362350
17    1   6 1.025 0.625815
```

Listing 32:

```
> 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 33:

```
> nrow(trim)

[1] 20000
```

Listing 34:

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

[1] 19000
```

Listing 35:

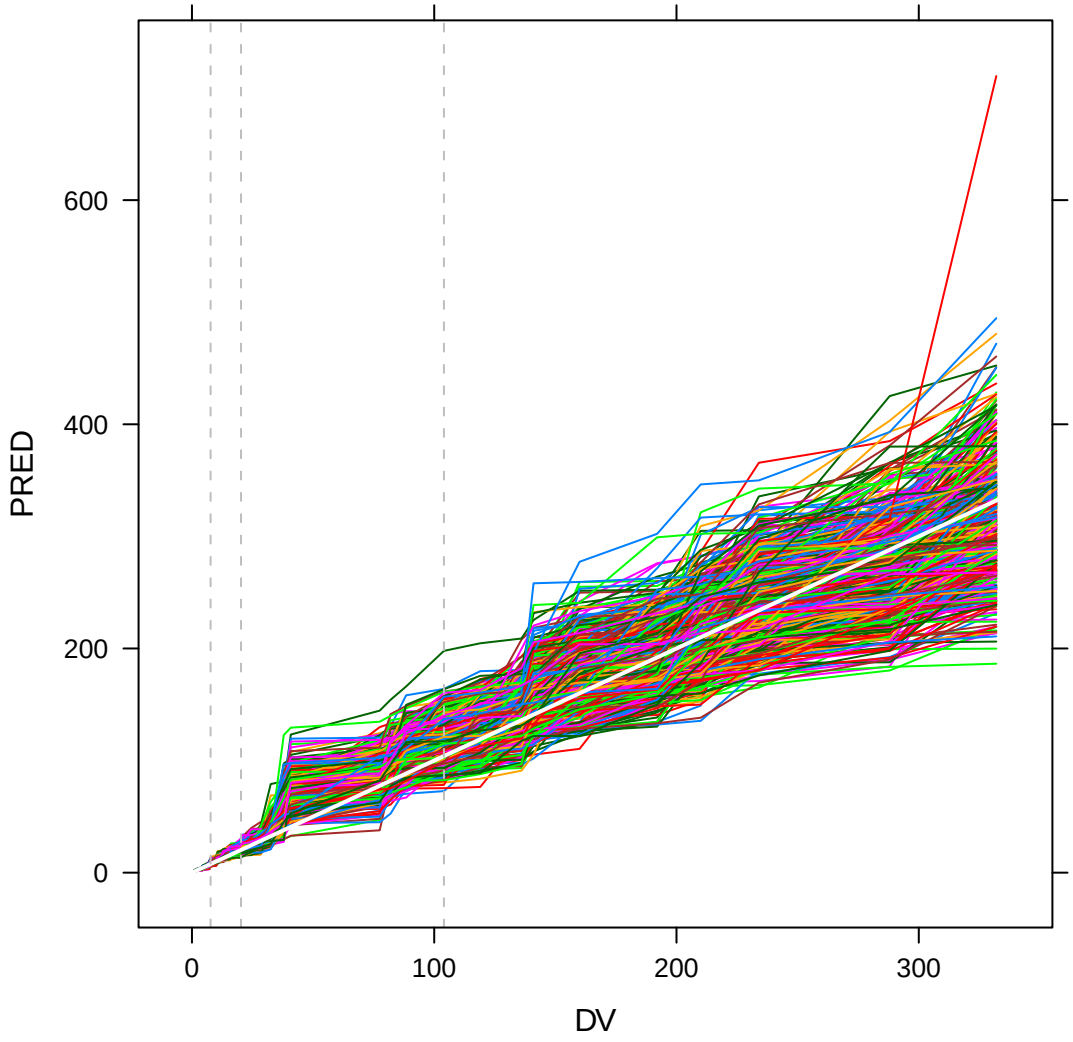
```
> head(trim)

    SIM   DV   PRED
501   1 1.13 1.9653
502   2 1.13 1.5989
503   3 1.13 1.4754
504   4 1.13 1.4074
505   5 1.13 1.3787
506   6 1.13 1.4753
```

Listing 36:

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



We also show densityplots of predictions at those quartiles.

Listing 37:

```
> head(trim)

    SIM   DV   PRED
501   1 1.13 1.9653
502   2 1.13 1.5989
503   3 1.13 1.4754
504   4 1.13 1.4074
505   5 1.13 1.3787
506   6 1.13 1.4753
```

Listing 38:

```
> quantile(trim$DV)

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

Listing 39:

```
> 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 40:

```
> 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  7.950000 20.2500 100.10000
2   1     PRED 10.329750 22.8675  91.61825
3   2       DV  7.950000 20.2500 100.10000
4   2     PRED 10.241500 23.4225  97.26175
5   3       DV  7.950000 20.2500 100.10000
6   3     PRED  8.081437 20.0330 106.59750
```

Listing 41:

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

```
  variable quartile     value
1       DV     X25.  7.950000
2     PRED     X25. 10.329750
3       DV     X25.  7.950000
4     PRED     X25. 10.241500
5       DV     X25.  7.950000
6     PRED     X25.  8.081437
```

Listing 42:

```
> levels(molt$quartile)

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

Listing 43:

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

  variable       quartile     value
1       DV first quartile  7.950000
2     PRED first quartile 10.329750
3       DV first quartile  7.950000
4     PRED first quartile 10.241500
5       DV first quartile  7.950000
6     PRED first quartile  8.081437
```

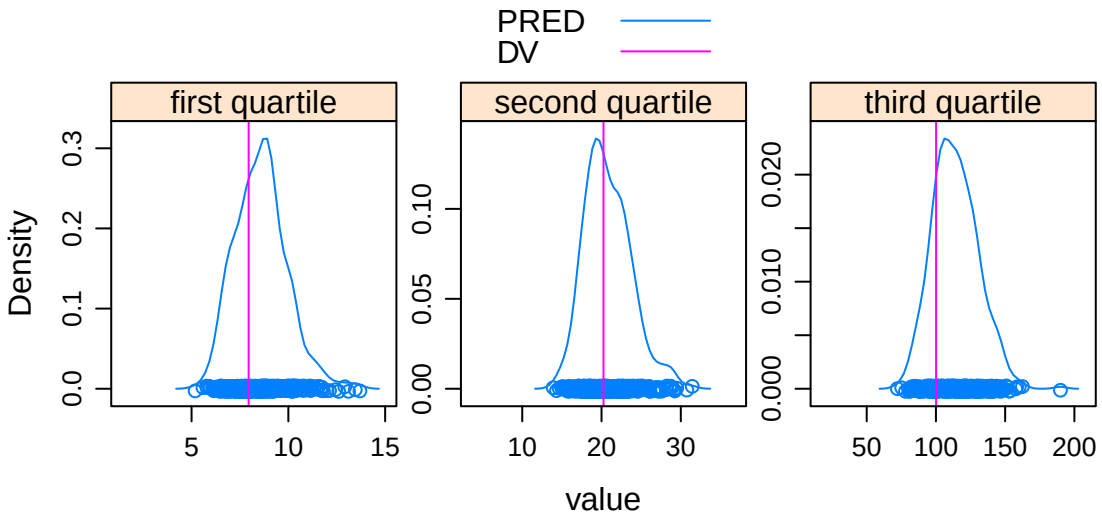
Listing 44:

```
> levels(molt$variable)

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

Listing 45:

```
> 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
+       )
+ )
```



4 Bootstrap Estimates of Parameter Uncertainty

4.1 Create directories.

Listing 46:

```
> getwd()
```

```
[1] "/Users/timb/project/metrum-mifuns/inst/sample/script"
```

Listing 47:

```
> dir.create('../nonmem/1005.boot')
> dir.create('../nonmem/1005.boot/data')
> dir.create('../nonmem/1005.boot/ctl')
```

4.2 Create replicate control streams.

Listing 48:

```
> t <- metaSub(
+      clear(readLines('../nonmem/ctl/1005.ctl'),';.+',fixed=FALSE),
+      names=1:300,
+      pattern=c(
+          '1005',
+          '../../data/derived/phase1.csv',
+          '$COV',
+          '$TABLE'
+      ),
+      replacement=c(
+          '*',
+          '../data/*.csv',
+          ';$COV',
+          ';$TABLE'
+     ),
+     fixed=TRUE,
+     out='../nonmem/1005.boot/ctl',
+     suffix='.ctl'
+  )
```

4.3 Create replicate data sets by resampling original.

Listing 49:

```
>  bootset <- read.csv('../data/derived/phase1.csv')
>  r <- resample(
+       bootset,
+       names=1:300,
+       key='ID',
+       rekey=TRUE,
+       out='../nonmem/1005.boot/data',
+       stratify='SEX'
+  )
```

4.4 Run bootstrap models.

To force a re-run of bootstraps, delete log.csv.

Listing 50:

```
> if(!file.exists('../nonmem/1005.boot/CombRunLog.csv'))NONR(
+      run=1:300,
+      command=command,
+      project='../nonmem/1005.boot/',
+      boot=TRUE,
+      nice=TRUE,
+      streams='../nonmem/1005.boot/ctl'
+ )
> getwd()
```

```
[1] "/Users/timb/project/metrum-mifuns/inst/sample/script"
```

4.5 Summarize bootstrap models.

When the bootstraps are complete, we return here and summarize. If you do not have time for bootstraps, use read.csv() on ../nonmem/1005.boot/log.csv.

Listing 51:

```
> #wait for bootstraps to finish
> #while(!(all(file.exists(paste(sep='','../nonmem/1005.boot/',1:300,'.boot
    /',1:300,'.lst'))))){}
> if(file.exists('../nonmem/1005.boot/log.csv')){
+     boot <- read.csv('../nonmem/1005.boot/log.csv',as.is=TRUE)
+ }else{
+     boot <- rlog(
+       run=1:300,
+       project='../nonmem/1005.boot',
+       boot=TRUE,
+       append=FALSE,
+       tool='nm7'
+     )
+     write.csv(boot, '../nonmem/1005.boot/log.csv')
+ }
> head(boot)
```

```
  X tool run parameter   moment
1 1  nm7   1      prob     text
2 2  nm7   1       min   status
3 3  nm7   1       ofv  minimum
4 4  nm7   1    THETA1 estimate
5 5  nm7   1    THETA1     prse
6 6  nm7   1    THETA2 estimate
                                                value
1 1 phase1 2 CMT like 1004 but diff. initial on V3
2                                                 0
3                                  2760.84241850239
4                                           7.98893
```

```
5                                          <NA>
6                                        19.892
```

Listing 52:

```
> unique(boot$parameter)

 [1] "prob"     "min"      "ofv"      "THETA1"   "THETA2"   "THETA3"
 [7] "THETA4"   "THETA5"   "THETA6"   "THETA7"   "OMEGA1.1" "OMEGA2.1"
[13] "OMEGA2.2" "OMEGA3.1" "OMEGA3.2" "OMEGA3.3" "SIGMA1.1"
```

Listing 53:

```
> text2decimal(unique(boot$parameter))

 [1]  NA  NA  NA 1.0 2.0 3.0 4.0 5.0 6.0 7.0 1.1 2.1 2.2 3.1 3.2 3.3 1.1
```

Listing 54:

```
> boot$X <- NULL
```

It looks like we have 14 estimated parameters. We will map them to the original control stream.

Listing 55:

```
> boot <- boot[!is.na(text2decimal(boot$parameter)),]
> head(boot)

  tool run parameter   moment     value
4  nm7   1    THETA1 estimate   7.98893
5  nm7   1    THETA1     prse      <NA>
6  nm7   1    THETA2 estimate    19.892
7  nm7   1    THETA2     prse      <NA>
8  nm7   1    THETA3 estimate 0.0650249
9  nm7   1    THETA3     prse      <NA>
```

Listing 56:

```
> unique(boot$moment)

[1] "estimate" "prse"
```

Listing 57:

```
> unique(boot$value[boot$moment=='prse'])

[1] NA
```

prse, and therefore moment, is noninformative for these bootstraps.

Listing 58:

```
> boot <- boot[boot$moment=='estimate',]
> boot$moment <- NULL
> unique(boot$tool)
```

```
[1] "nm7"
```

Listing 59:

```
> boot$tool <- NULL
> head(boot)
```

```
   run parameter     value
4    1    THETA1   7.98893
6    1    THETA2    19.892
8    1    THETA3 0.0650249
10   1    THETA4   3.35627
12   1    THETA5   123.566
14   1    THETA6   1.18258
```

Listing 60:

```
> unique(boot$value[boot$parameter %in% c('OMEGA2.1','OMEGA3.1','OMEGA3.2')])
```

```
[1] "0"
```

Listing 61:

```
> unique(boot$parameter[boot$value=='0'])
```

```
[1] "OMEGA2.1" "OMEGA3.1" "OMEGA3.2"
```

Off-diagonals (and only off-diagonals) are noninformative.

Listing 62:

```
> boot <- boot[!boot$value=='0',]
> any(is.na(as.numeric(boot$value)))
```

```
[1] FALSE
```

Listing 63:

```
> boot$value <- as.numeric(boot$value)
> head(boot)
```

```
   run parameter       value
4    1    THETA1   7.9889300
6    1    THETA2  19.8920000
8    1    THETA3   0.0650249
10   1    THETA4   3.3562700
12   1    THETA5 123.5660000
14   1    THETA6   1.1825800
```

4.6 Restrict data to 95 percentiles.

We did 300 runs. Min and max are strongly dependent on number of runs, since with an unbounded distribution, (almost) any value is possible with enough sampling. We clip to the 95 percentiles, to give distributions that are somewhat more scale independent.

Listing 64:

```
> boot <- inner(
+       boot,
+       preserve='run',
+       id.var='parameter',
+       measure.var='value'
+ )
> head(boot)
```

```
  run parameter       value
1   1    THETA1   7.9889300
2   1    THETA2  19.8920000
3   1    THETA3   0.0650249
4   1    THETA4   3.3562700
5   1    THETA5 123.5660000
6   1    THETA6   1.1825800
```

Listing 65:

```
> any(is.na(boot$value))
```

```
[1] TRUE
```

Listing 66:

```
> boot <- boot[!is.na(boot$value),]
```

4.7 Recover parameter metadata from a specially-marked control stream.

We want meaningful names for our parameters. Harvest these from a reviewed control stream.

Listing 67:

```
> stream <- readLines('../nonmem/ctl/1005.ctl')
> tail(stream)
```

```
[1] ";<parameter name='SIGMA1.1' label='$\\sigma^{1.1}prop$'>proportional error</
    parameter>"
[2] ""
[3] "$ESTIMATION MAXEVAL=9999 PRINT=5 NOABORT METHOD=1 INTER MSFO=./1005.msf"
[4] "$COV PRINT=E"
[5] "$TABLE NOPRINT FILE=./1005.tab ONEHEADER ID AMT TIME EVID PRED IPRE CWRES"
[6] "$TABLE NOPRINT FILE=./1005par.tab ONEHEADER ID TIME CL Q V2 V3 KA ETA1 ETA2
    ETA3"
```

Listing 68:

```
> doc <- ctl2xml(stream)
> doc
```

```
 [1] "<document>"
 [2] "<parameter name='THETA1' latex='$\\theta_1$' unit='$L/h$'    label='CL/F'
     model='$CL/F \\sim \\theta_6^{MALE} * (WT/70)^{\\theta_7}$'>clearance</
     parameter>"
 [3] "<parameter name='THETA2' latex='$\\theta_2$' unit='$L$'      label='Vc/F'
     model='$Vc/F \\sim (WT/70)^{1}$'   >central volume</parameter>"
 [4] "<parameter name='THETA3' latex='$\\theta_3$' unit='$h^{-1}$' label='Ka'
                                        >absorption constant</parameter>"
 [5] "<parameter name='THETA4' latex='$\\theta_4$' unit='$L/h$'    label='Q/F'
                                       >intercompartmental clearance</parameter>"
 [6] "<parameter name='THETA5' latex='$\\theta_5$' unit='$L$'      label='Vp/F'
                                      >peripheral volume</parameter>"
 [7] "<parameter name='THETA6' latex='$\\theta_6$'                 label='Male.CL'
                                    >male effect on clearance</parameter>"
 [8] "<parameter name='THETA7' latex='$\\theta_7$'                 label='WT.CL'
                                     >weight effect on clearance</parameter>"
 [9] "<parameter name='OMEGA1.1' label='$\\Omega^{1.1}CL/F$'>interindividual
     variability on clearance</parameter>"
[10] "<parameter name='OMEGA2.2' label='$\\Omega^{2.2}Vc/F$'>interindividual
    variability on central volume</parameter>"
[11] "<parameter name='OMEGA3.3' label='$\\Omega^{3.3}Ka$'>interindividual
    variability on Ka</parameter>"
[12] "<parameter name='SIGMA1.1' label='$\\sigma^{1.1}prop$'>proportional error</
    parameter>"
[13] "</document>"
```

Listing 69:

```
> params <- unique(boot[,'parameter',drop=FALSE])
> params$defs <- lookup(params$parameter,within=doc)
> params$labels <- lookup(params$parameter,within=doc,as='label')
> params
```

	parameter	defs	labels
1	THETA1	clearance	CL/F
2	THETA2	central volume	Vc/F
3	THETA3	absorption constant	Ka
4	THETA4	intercompartmental clearance	Q/F
5	THETA5	peripheral volume	Vp/F
6	THETA6	male effect on clearance	Male.CL
7	THETA7	weight effect on clearance	WT.CL
8	OMEGA1.1	interindividual variability on clearance	$\\Omega^{1.1}CL/F$
9	OMEGA2.2	interindividual variability on central volume	$\\Omega^{2.2}Vc/F$
10	OMEGA3.3	interindividual variability on Ka	$\\Omega^{3.3}Ka$
11	SIGMA1.1	proportional error	$\\sigma^{1.1}prop$

Listing 70:

```
> boot$parameter <- lookup(boot$parameter,within=doc,as='label')
> head(boot)
```

```
  run parameter       value
1   1      CL/F   7.9889300
2   1      Vc/F  19.8920000
3   1        Ka   0.0650249
4   1       Q/F   3.3562700
5   1      Vp/F 123.5660000
6   1   Male.CL   1.1825800
```

4.8 Create covariate plot.

Now we make a covariate plot for clearance. We will normalize clearance by its median (we also could have used the model estimate). We need to take cuts of weight, since we can only really show categorically-constrained distributions. Male effect is already categorical. I.e, the reference individual has median clearance, is female, and has median weight.

4.8.1 Recover original covariates for guidance.

Listing 71:

```
> covariates <- read.csv('../data/derived/phase1.csv',na.strings='.')
> head(covariates)
```

```
     C ID TIME SEQ EVID  AMT    DV SUBJ HOUR TAFD  TAD LDOS MDV HEIGHT WEIGHT
1    C  1 0.00   0    0   NA 0.000    1 0.00 0.00   NA   NA   0    174   74.2
2 <NA>  1 0.00   1    1 1000    NA    1 0.00 0.00 0.00 1000   1    174   74.2
3 <NA>  1 0.25   0    0   NA 0.363    1 0.25 0.25 0.25 1000   0    174   74.2
4 <NA>  1 0.50   0    0   NA 0.914    1 0.50 0.50 0.50 1000   0    174   74.2
5 <NA>  1 1.00   0    0   NA 1.120    1 1.00 1.00 1.00 1000   0    174   74.2
6 <NA>  1 2.00   0    0   NA 2.280    1 2.00 2.00 2.00 1000   0    174   74.2
  SEX  AGE DOSE FED SMK DS CRCN predose zerodv
1   0 29.1 1000   1   0  0 83.5       1      1
2   0 29.1 1000   1   0  0 83.5       0      0
3   0 29.1 1000   1   0  0 83.5       0      0
4   0 29.1 1000   1   0  0 83.5       0      0
5   0 29.1 1000   1   0  0 83.5       0      0
6   0 29.1 1000   1   0  0 83.5       0      0
```

Listing 72:

```
> with(covariates,constant(WEIGHT,within=ID))
```

```
[1] TRUE
```

Listing 73:

```
> covariates <- unique(covariates[,c('ID','WEIGHT')])
> head(covariates)
```

```
   ID WEIGHT
1   1   74.2
16  2   80.3
31  3   94.2
46  4   85.2
61  5   82.8
76  6   63.9
```

Listing 74:

```
> covariates$WT <- as.numeric(covariates$WEIGHT)
> wt <- median(covariates$WT)
> wt
```

```
[1] 81
```

Listing 75:

```
> range(covariates$WT)
```

```
[1]  61 117
```

4.8.2 Reproduce the control stream submodel for selective cuts of a continuous covariate.

In the model we normalized by 70 kg, so that cut will have null effect. Let's try 65, 75, and 85 kg. We have to make a separate column for each cut, which is a bit of work. Basically, we make two more copies of our weight effect columns, and raise our normalized cuts to those powers, effectively reproducing the submodel from the control stream.

Listing 76:

```
> head(boot)
```

```
  run parameter       value
1   1      CL/F   7.9889300
2   1      Vc/F  19.8920000
3   1        Ka   0.0650249
4   1       Q/F   3.3562700
5   1      Vp/F 123.5660000
6   1   Male.CL   1.1825800
```

Listing 77:

```
> unique(boot$parameter)
```

```
 [1] "CL/F"                "Vc/F"                "Ka"
 [4] "Q/F"                 "Vp/F"                "Male.CL"
 [7] "WT.CL"               "$\\Omega^{1.1}CL/F$" "$\\Omega^{2.2}Vc/F$"
[10] "$\\Omega^{3.3}Ka$"   "$\\sigma^{1.1}prop$"
```

Listing 78:

```
> clearance <- boot[boot$parameter %in% c('CL/F','WT.CL','Male.CL'),]
> head(clearance)
```

```
   run parameter    value
1    1      CL/F 7.988930
6    1   Male.CL 1.182580
7    1     WT.CL 1.308790
12   2      CL/F 7.636730
17   2   Male.CL 0.956565
18   2     WT.CL 2.369810
```

Listing 79:

```
> frozen <- data.frame(cast(clearance,run ~ parameter),check.names=FALSE)
> head(frozen)
```

```
  run    CL/F  Male.CL   WT.CL
1   1 7.98893 1.182580 1.30879
2   2 7.63673 0.956565 2.36981
3   3 9.15198 0.937231 1.88593
4   4 9.56138 1.028670 1.47186
5   5 8.36964 0.914796 1.97656
6   6 9.09701 1.079030 1.16319
```

Listing 80:

```
> frozen$WT.CL65 <- (65/70)**frozen$WT.CL
> frozen$WT.CL75 <- (75/70)**frozen$WT.CL
> frozen$WT.CL85 <- (85/70)**frozen$WT.CL
```

4.8.3 Normalize key parameter

Listing 81:

```
> cl <- median(boot$value[boot$parameter=='CL/F'])
> cl
```

```
[1] 8.56139
```

Listing 82:

```
> head(frozen)
```

```
  run    CL/F  Male.CL   WT.CL   WT.CL65  WT.CL75  WT.CL85
1   1 7.98893 1.182580 1.30879 0.9075635 1.094499 1.289313
2   2 7.63673 0.956565 2.36981 0.8389352 1.177625 1.584253
3   3 9.15198 0.937231 1.88593 0.8695648 1.138960 1.442193
4   4 9.56138 1.028670 1.47186 0.8966618 1.106883 1.330787
5   5 8.36964 0.914796 1.97656 0.8637440 1.146104 1.467795
6   6 9.09701 1.079030 1.16319 0.9174092 1.083560 1.253376
```

Listing 83:

```
> frozen[['CL/F']] <- frozen[['CL/F']]/cl
> head(frozen)
```

```
  run      CL/F  Male.CL   WT.CL   WT.CL65  WT.CL75  WT.CL85
1   1 0.9331347 1.182580 1.30879 0.9075635 1.094499 1.289313
2   2 0.8919965 0.956565 2.36981 0.8389352 1.177625 1.584253
3   3 1.0689830 0.937231 1.88593 0.8695648 1.138960 1.442193
4   4 1.1168023 1.028670 1.47186 0.8966618 1.106883 1.330787
5   5 0.9776029 0.914796 1.97656 0.8637440 1.146104 1.467795
6   6 1.0625623 1.079030 1.16319 0.9174092 1.083560 1.253376
```

Listing 84:

```
> frozen$WT.CL <- NULL
> molten <- melt(frozen,id.var='run',na.rm=TRUE)
> head(molten)
```

```
  run variable     value
1   1     CL/F 0.9331347
2   2     CL/F 0.8919965
3   3     CL/F 1.0689830
4   4     CL/F 1.1168023
5   5     CL/F 0.9776029
6   6     CL/F 1.0625623
```

4.8.4 Plot.

Now we plot. We reverse the variable factor to give us top-down layout of strips.

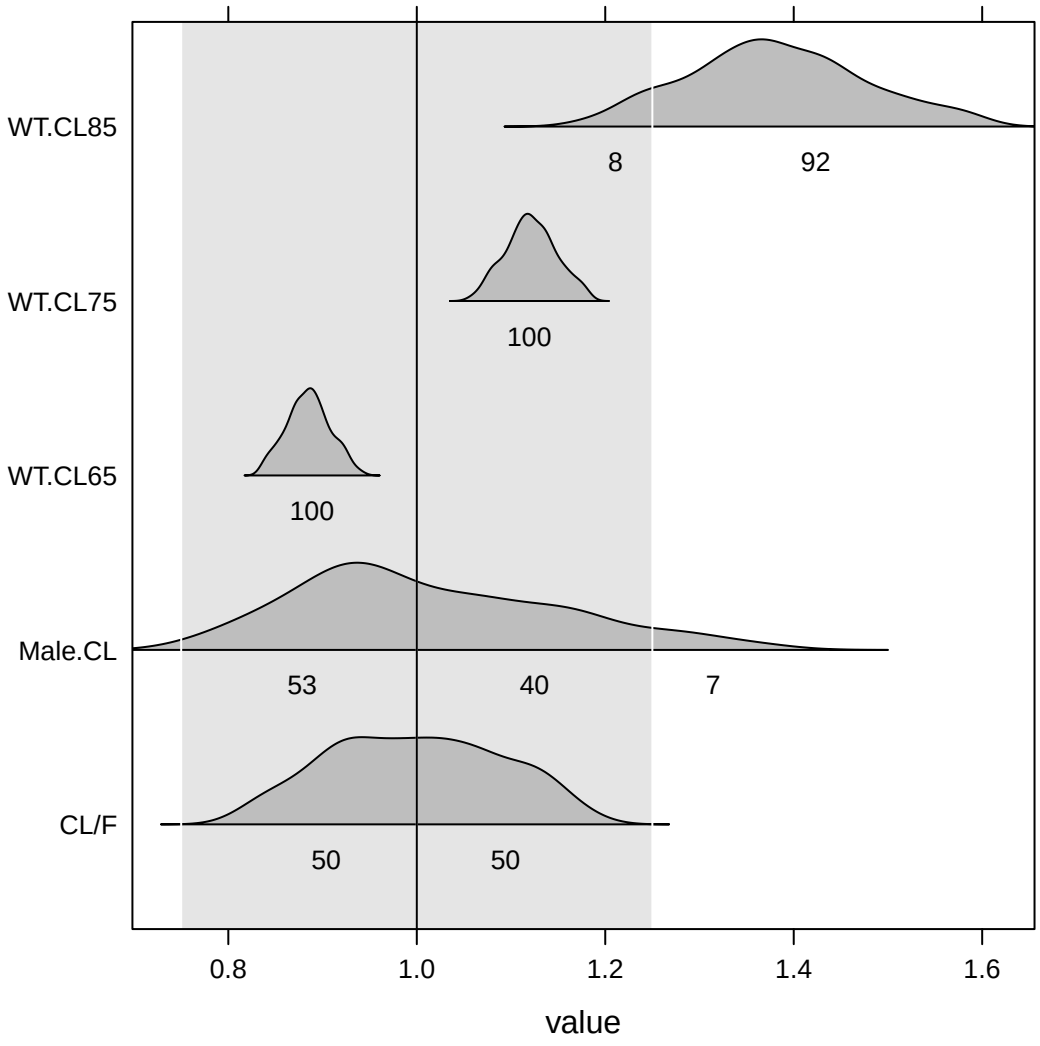
Listing 85:

```
> levels(molten$variable)
```

```
[1] "CL/F"    "Male.CL" "WT.CL65" "WT.CL75" "WT.CL85"
```

Listing 86:

```
> molten$variable <- factor(molten$variable,levels=rev(levels(molten$variable)))
> print(stripplot(variable ~ value,molten,panel=panel.covplot))
```



4.8.5 Summarize

We see that clearance is estimated with good precision. Ignoring outliers, there is not much effect on clearance of being male, relative to female. Increasing weight is associated with increasing clearance. There is a 79 percent probability that an 85 kg person will have at least 25 percent greater clearance than a 70 kg person.

5 Parameter Table

Listing 87:

```
> library(Hmisc)
> tab <- partab(1005,'../nonmem',tool='nm7',as=c('label','latex','model','estimate
    ','unit','prse','se'))
> tab$estimate <- as.character(signif(as.numeric(tab$estimate),3))
> tab$estimate <- ifelse(is.na(tab$unit),tab$estimate,paste(tab$estimate, tab$unit
    ))
> tab$unit <- NULL
> tab$label <- ifelse(is.na(tab$latex),tab$label,paste(tab$label, ' (',tab$latex
    ,')',sep=''))
> tab$latex <- NULL
> names(tab)[names(tab)=='label'] <- 'parameter'
> tab$root <- signif(sqrt(exp(as.numeric(tab$estimate))-1),3)
> tab$estimate <- ifelse(contains('Omega|sigma',tab$parameter),paste(tab$estimate
    ,' (\\%CV=',tab$root*100,')',sep=''),tab$estimate)
> tab$root <- NULL
> #offdiag <- contains('2.1',tab$parameter)
> #tab$estimate[offdiag] <- text2decimal(tab$estimate[offdiag])
> #omegablock <- text2decimal(tab$estimate[contains('Omega..(1|2)',tab$parameter)
    ])
> #cor <- signif(half(cov2cor(as.matrix(as.halfmatrix(omegablock))))[[2]],3)
> #tab$estimate[offdiag] <- paste(sep='',tab$estimate[offdiag],' (COR=',cor,')')
> tab$model[is.na(tab$model)] <- ''
> #boot <- rlog(1:300,project='../nonmem/1005.boot',tool='nm7',boot=TRUE)
> boot <- read.csv('../nonmem/1005.boot/log.csv',as.is=TRUE)
> boot <- boot[boot$moment=='estimate',]
> boot <- data.frame(cast(boot,... ~ moment))
> boot[] <- lapply(boot,as.character)
> boot <- boot[contains('THETA|OMEGA|SIGMA',boot$parameter),c('parameter','
    estimate')]
> boot$estimate <- as.numeric(boot$estimate)
> boot <- data.frame(cast(boot,parameter ~ .,value='estimate',fun=function(x)list(
    lo=as.character(signif(quantile(x,probs=0.05),3)),hi=as.character(signif(
    quantile(x,probs=0.95),3)))))
> boot$CI <- with(boot, paste(sep='','(',lo,',',hi,')'))
> names(boot)[names(boot)=='parameter'] <- 'name'
> tab <- stableMerge(tab,boot[,c('name','CI')])
> tab$name <- NULL
> tab$se <- NULL
```

Table 1: Parameter Estimates from Population Pharmacokinetic Model Run 1005

parameter	model	estimate	prse	CI
CL/F (θ_1)	$CL/F \sim \theta_6^{MALE} * (WT/70)^{\theta_7}$	8.58 L/h	9.51	(7.14,9.89)
Vc/F (θ_2)	$Vc/F \sim (WT/70)^1$	21.6 L	9.33	(18.5,25.4)
Ka (θ_3)		0.0684 h^{-1}	8.04	(0.0586,0.0793)
Q/F (θ_4)		3.78 L/h	13.5	(3.03,4.83)
Vp/F (θ_5)		107 L	15.7	(85.7,148)
Male.CL (θ_6)		0.999	13.7	(0.799,1.31)
WT.CL (θ_7)		1.67	21.9	(1.03,2.34)
$\Omega^{1.1}CL/F$		0.196 (%CV=46.5)	23.1	(0.115,0.26)
$\Omega^{2.2}Vc/F$		0.129 (%CV=37.1)	30.4	(0.0623,0.181)
$\Omega^{3.3}Ka$		0.107 (%CV=33.6)	25.2	(0.0638,0.157)
$\sigma^{1.1}prop$		0.0671 (%CV=26.3)	11.4	(0.055,0.0796)