MIfuns Sample Script

Phase I Modeling

June 17, 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 4.0.8 loaded
Installing SIGCHLD signal handler...Done.
```

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.

Listing 6:

```
> t <- metaSub(
+      as.filename('../nonmem/ctl/1005.ctl'),
+      names=1105,
+      pattern=c(
+          '\\$THETA[^$]+',
+          '\\$OMEGA[^$]+',
+          '\\$SIGMA[^$]+',
+          '\\$EST[^$]+',
+          '\\$COV',
+          '\\$TABLE.*'
+      ),
+      replacement=c(
+          '$MSFI=../1005/1005.msf\n',
+          ';$OMEGA\n',
+          ';$SIGMA\n',
+          '$SIMULATION ONLYSIM (1968) SUBPROBLEMS=500\n',
+          ';$COV',
+          '$TABLE DV NOHEADER NOPRINT FILE=./*.tab FORWARD NOAPPEND\n'
+     ),
+     fixed=FALSE,
+     out='../nonmem/ctl',
+     suffix='.ctl'
+ )
```

3.2 Run the simulation.

This run makes the predictions (simulations).

Listing 7:

```
> 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 8:

```
> 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 9:

```
> 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 10:

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

```
[1] 550
```

Listing 11:

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

```
[1] 275000
```

Listing 12:

```
> 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 13:

```
> 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 14:

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

```
[1] 275000
```

Listing 15:

```
> length(pred)
```

```
[1] 275000
```

3.5 Combine the original data and the simulation data.

Listing 16:

```
> 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 17:

```
> 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 18:

```
> library(reshape)
> 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 19:

```
> 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 20:

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

Now we cast, ignoring time.

Listing 21:

```
> 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 22:

```
> 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
```

3.6.2 Aggregate data across subjects, within simulations.

Our predictions have central tendencies, which can vary by SIM. Thus, our metrics as well have central tendencies that vary by SIM. We want to represent the variability across SIMS by aggregating within SIM. That means aggregating across subjects, within SIMS. There are many aggregation strategies, but we choose quantiles for a non-parametric result. Quantiles that 'clip' the tails of the distribution offer robustness against number of SIMS (i.e., results less dependent on number of sims). Within each SIM, let's find for each metric the 5th, 50th, and 95th percentile. We also want to do this for the original data set (requires some minor rearrangement).

Listing 23:

```
> 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 24:

```
> quants <- data.frame(cast(metr,SIM + metric + variable  ~  .,fun=quantile,probs=c
    (0.05,0.50,0.95)))
> head(quants,10)
```

```
   SIM metric variable       X5.     X50.      X95.
1    1    min       DV 0.3054500  2.1450  36.0750
2    1    min     PRED 0.0976828  2.3129  29.6127
3    1    med       DV 1.5860000 20.2500 290.2000
4    1    med     PRED 2.2552400 22.8675 304.0180
5    1    max       DV 3.0855000 40.7000 634.2500
6    1    max     PRED 4.4729900 47.2865 579.6585
7    2    min       DV 0.3054500  2.1450  36.0750
8    2    min     PRED 0.0949232  2.8080  32.3266
9    2    med       DV 1.5860000 20.2500 290.2000
10   2    med     PRED 1.6609825 23.4225 263.8535
```

Note, again, that DV quantiles are invariant across SIMS.

3.6.3 Reformat data for bivariate display.

We now have a lot of display options. The simplest is to plot DV PRED for each quantile and metric. Requires slight rearrangement.

Listing 25:

```
> molten <- melt(quants, measure.var=c('X5.','X50.','X95.'),variable_name='quant')
> head(molten)
```

```
  SIM metric variable quant     value
1   1    min       DV   X5. 0.3054500
2   1    min     PRED   X5. 0.0976828
3   1    med       DV   X5. 1.5860000
4   1    med     PRED   X5. 2.2552400
5   1    max       DV   X5. 3.0855000
6   1    max     PRED   X5. 4.4729900
```

Listing 26:

```
> frozen <- data.frame(cast(molten, SIM + metric + quant  ~  variable))
> head(frozen)
```

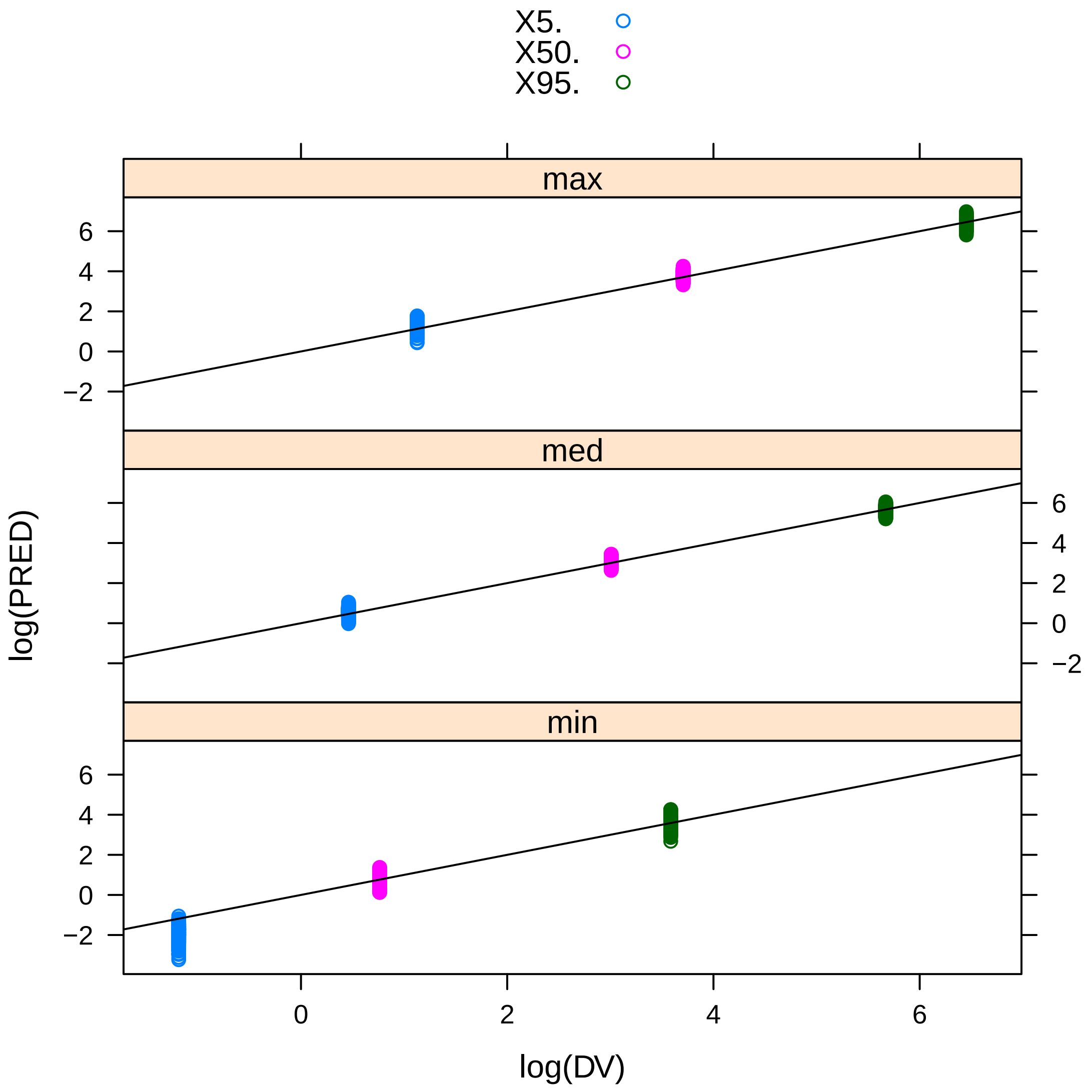
```
  SIM metric quant        DV        PRED
1   1    min   X5.   0.30545   0.0976828
2   1    min  X50.   2.14500   2.3129000
3   1    min  X95.  36.07500  29.6127000
```

```
4   1    med   X5.   1.58600   2.2552400
5   1    med  X50.  20.25000  22.8675000
6   1    med  X95. 290.20000 304.0180000
```

3.6.4 Bivariate display of within-simulation aggregate metrics.

Listing 27:

```
> print(xyplot(
+       log(PRED) ~ log(DV)|metric,
+       frozen,
+       groups=quant,
+       layout=c(1,3),
+       auto.key=TRUE,
+       panel=function(...){
+               panel.xyplot(...)
+               panel.abline(a=0,b=1)
+       }
+ ))
```



3.6.5 Univariate displays.

For a better view of the distributions, however, we can work with single-axis plot functions, using the molten data. For faster and clearer plotting, we remove duplicates of DV.

3.6.6 Classic stripplot

Listing 28:

```
> head(molten)

  SIM metric variable quant      value
1   1    min       DV   X5. 0.3054500
2   1    min     PRED   X5. 0.0976828
3   1    med       DV   X5. 1.5860000
4   1    med     PRED   X5. 2.2552400
5   1    max       DV   X5. 3.0855000
6   1    max     PRED   X5. 4.4729900
```

Listing 29:

```
> molten$SIM <- NULL
> table(molten$variable)

  DV PRED
4500 4500
```

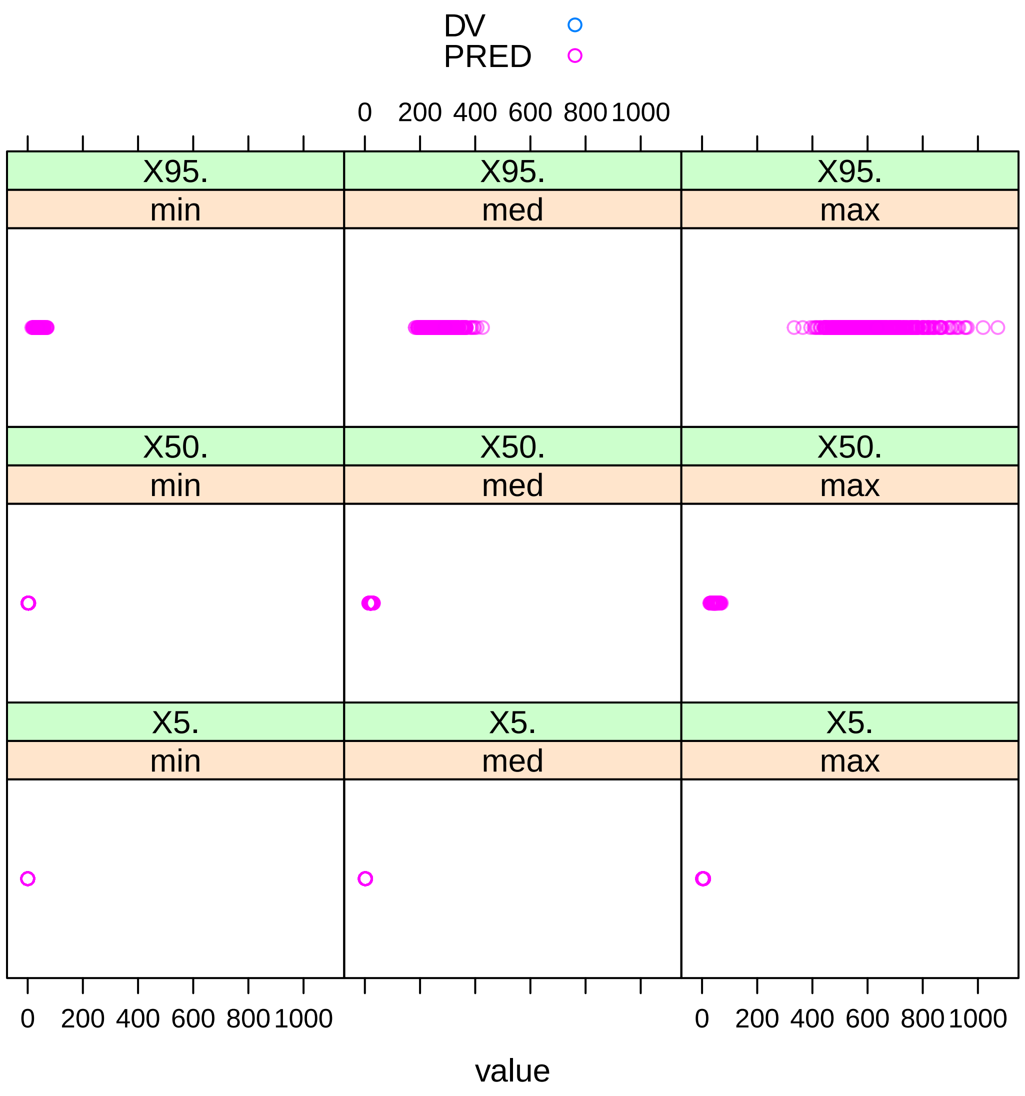
Listing 30:

```
> molten <- molten[!(duplicated(molten[,c('metric','variable','quant')]) &
    molten$variable=='DV'),]
> table(molten$variable)

  DV PRED
   9 4500
```

Listing 31:

```
> library(grid)
> print(stripplot(
+         ~ value|metric+quant,
+        molten,
+        groups=variable,
+        horizontal=TRUE,
+        auto.key=TRUE,
+        panel=panel.superpose,
+        alpha=0.5,
+        panel.groups=panel.stripplot
+ ))
```

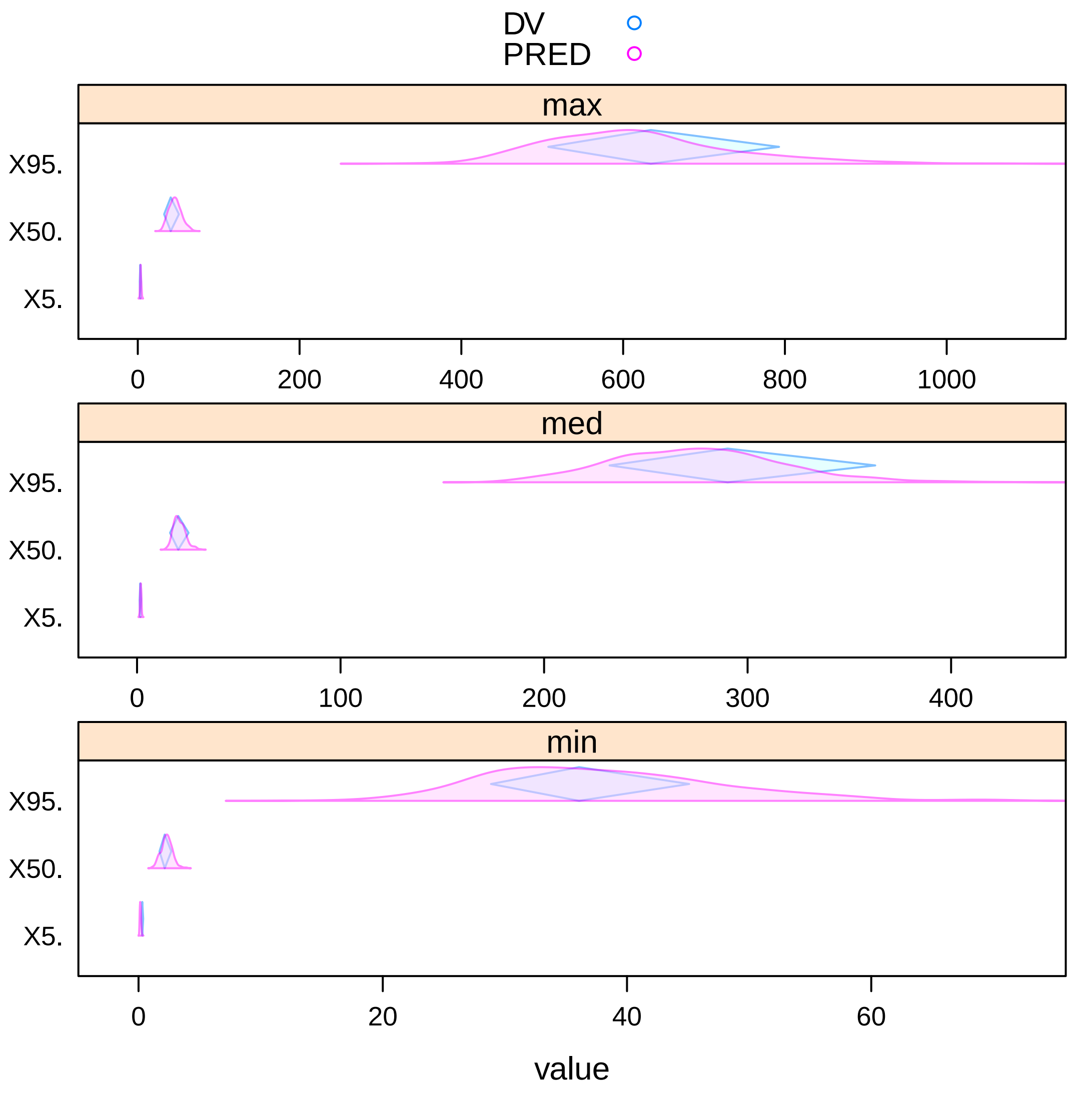


3.6.7 Diamondback: reference regions on density strips

Here we show the distribution data as density strips, and indicate reference regions around the point estimates. Here is one option. Also try swapping ‘quant’ and ‘metric’.

Listing 32:

```
> print(stripplot(
+       quant ~ value|metric,
+       molten,
+       groups=variable,
+       auto.key=TRUE,
+       panel=panel.stratify,
+       alpha=0.5,
+       layout=c(1,3),
+       scales=list(relation='free'),
+       panel.levels = function(x,y,group.number,col,col.line,fill,font,...){
+               if(group.number==1)for(d in seq(length.out=length(x))) panel.
    polygon(
+                       x=x[[d]]*c(0.8,1,1.25,1),
+                       y=y[[d]] + c(0.25,0,0.25,0.5),
+                       border=col.line,
+                       col=fill,
+                       ...
+               )
+               else panel.densitystrip(x=x,y=y,col=fill,border=col.line,...)
+       }
+ ))
```



4 Bootstrap Estimates of Parameter Uncertainty

4.1 Create directories.

Listing 33:

```
> getwd()
```

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

Listing 34:

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

4.2 Create replicate control streams.

Listing 35:

```
> 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 36:

```
>  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 37:

```
> 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 38:

```
> #boot <- read.csv('../nonmem/1005.boot/log.csv',as.is=TRUE)
> #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 estimate
4 4  nm7   1  OMEGA1.1 estimate
5 5  nm7   1  OMEGA1.1     prse
6 6  nm7   1  OMEGA2.1 estimate
                                         value
1 1 phase1 2 CMT like 1004 but diff. initial on V3
2                                            0
3                             2760.84241850239
```

```
4                                         0.188595
5                                             <NA>
6                                                0
```

Listing 39:

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

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

Listing 40:

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

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

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

Listing 41:

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

```
  X tool run parameter   moment    value
4 4  nm7   1  OMEGA1.1 estimate 0.188595
5 5  nm7   1  OMEGA1.1     prse     <NA>
6 6  nm7   1  OMEGA2.1 estimate        0
7 7  nm7   1  OMEGA2.1     prse     <NA>
8 8  nm7   1  OMEGA2.2 estimate 0.112992
9 9  nm7   1  OMEGA2.2     prse     <NA>
```

Listing 42:

```
> unique(boot$moment)
```

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

Listing 43:

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

```
[1] NA
```

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

Listing 44:

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

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

Listing 45:

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

```
    X run parameter     value
4   4   1  OMEGA1.1  0.188595
6   6   1  OMEGA2.1         0
8   8   1  OMEGA2.2  0.112992
10 10   1  OMEGA3.1         0
12 12   1  OMEGA3.2         0
14 14   1  OMEGA3.3 0.0854714
```

Listing 46:

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

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

Listing 47:

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

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

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

Listing 48:

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

```
[1] FALSE
```

Listing 49:

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

```
    X run parameter      value
4   4   1  OMEGA1.1  0.1885950
8   8   1  OMEGA2.2  0.1129920
14 14   1  OMEGA3.3  0.0854714
16 16   1  SIGMA1.1  0.0640717
18 18   1    THETA1  7.9889300
20 20   1    THETA2 19.8920000
```

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 50:

```
> boot$upper <- with(boot,reapply(value,INDEX=parameter,FUN=quantile,probs=0.975))
> boot$lower <- with(boot,reapply(value,INDEX=parameter,FUN=quantile,probs=0.025))
> nrow(boot)
```

```
[1] 3300
```

Listing 51:

```
> boot <- boot[with(boot, value < upper & value > lower),]
> nrow(boot)
```

```
[1] 3124
```

Listing 52:

```
> head(boot)
```

```
    X run parameter      value       upper       lower
4   4   1  OMEGA1.1  0.1885950  0.28599012  0.10657525
8   8   1  OMEGA2.2  0.1129920  0.19930930  0.05290951
14 14   1  OMEGA3.3  0.0854714  0.16658800  0.05277859
16 16   1  SIGMA1.1  0.0640717  0.08328707  0.05422743
18 18   1    THETA1  7.9889300 10.18006750  6.90182250
20 20   1    THETA2 19.8920000 26.29300250 17.74395750
```

Listing 53:

```
> boot$upper <- NULL
> boot$lower <- NULL
> head(boot)
```

```
    X run parameter      value
4   4   1  OMEGA1.1  0.1885950
8   8   1  OMEGA2.2  0.1129920
14 14   1  OMEGA3.3  0.0854714
16 16   1  SIGMA1.1  0.0640717
18 18   1    THETA1  7.9889300
20 20   1    THETA2 19.8920000
```

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 54:

```
> 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 55:

```
> 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 56:

```
> 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
4   OMEGA1.1      interindividual variability on clearance $\\Omega^{1.1}CL/F$
8   OMEGA2.2 interindividual variability on central volume $\\Omega^{2.2}Vc/F$
14  OMEGA3.3             interindividual variability on Ka   $\\Omega^{3.3}Ka$
16  SIGMA1.1                            proportional error $\\sigma^{1.1}prop$
18    THETA1                                     clearance                CL/F
20    THETA2                                central volume                Vc/F
22    THETA3                           absorption constant                  Ka
24    THETA4                      intercompartmental clearance             Q/F
26    THETA5                             peripheral volume                Vp/F
28    THETA6                      male effect on clearance             Male.CL
30    THETA7                    weight effect on clearance               WT.CL
```

Listing 57:

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

```
    X run           parameter      value
4   4   1 $\\Omega^{1.1}CL/F$  0.1885950
8   8   1 $\\Omega^{2.2}Vc/F$  0.1129920
14 14   1   $\\Omega^{3.3}Ka$  0.0854714
16 16   1 $\\sigma^{1.1}prop$  0.0640717
18 18   1                CL/F  7.9889300
20 20   1                Vc/F 19.8920000
```

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 58:

```
> 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 59:

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

```
[1] TRUE
```

Listing 60:

```
> 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 61:

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

```
[1] 81
```

Listing 62:

```
> 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 63:

```
> head(boot)

    X run           parameter      value
4   4   1 $\\Omega^{1.1}CL/F$  0.1885950
8   8   1 $\\Omega^{2.2}Vc/F$  0.1129920
14 14   1   $\\Omega^{3.3}Ka$  0.0854714
16 16   1 $\\sigma^{1.1}prop$  0.0640717
18 18   1                CL/F  7.9889300
20 20   1                Vc/F 19.8920000
```

Listing 64:

```
> unique(boot$parameter)

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

Listing 65:

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

    X run parameter    value
18 18   1      CL/F 7.988930
28 28   1   Male.CL 1.182580
30 30   1     WT.CL 1.308790
49 49   2      CL/F 7.636730
59 59   2   Male.CL 0.956565
61 61   2     WT.CL 2.369810
```

Listing 66:

```
> 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 67:

```
> 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 68:

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

```
[1] 8.56139
```

Listing 69:

```
> 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 70:

```
> 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 71:

```
> 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.

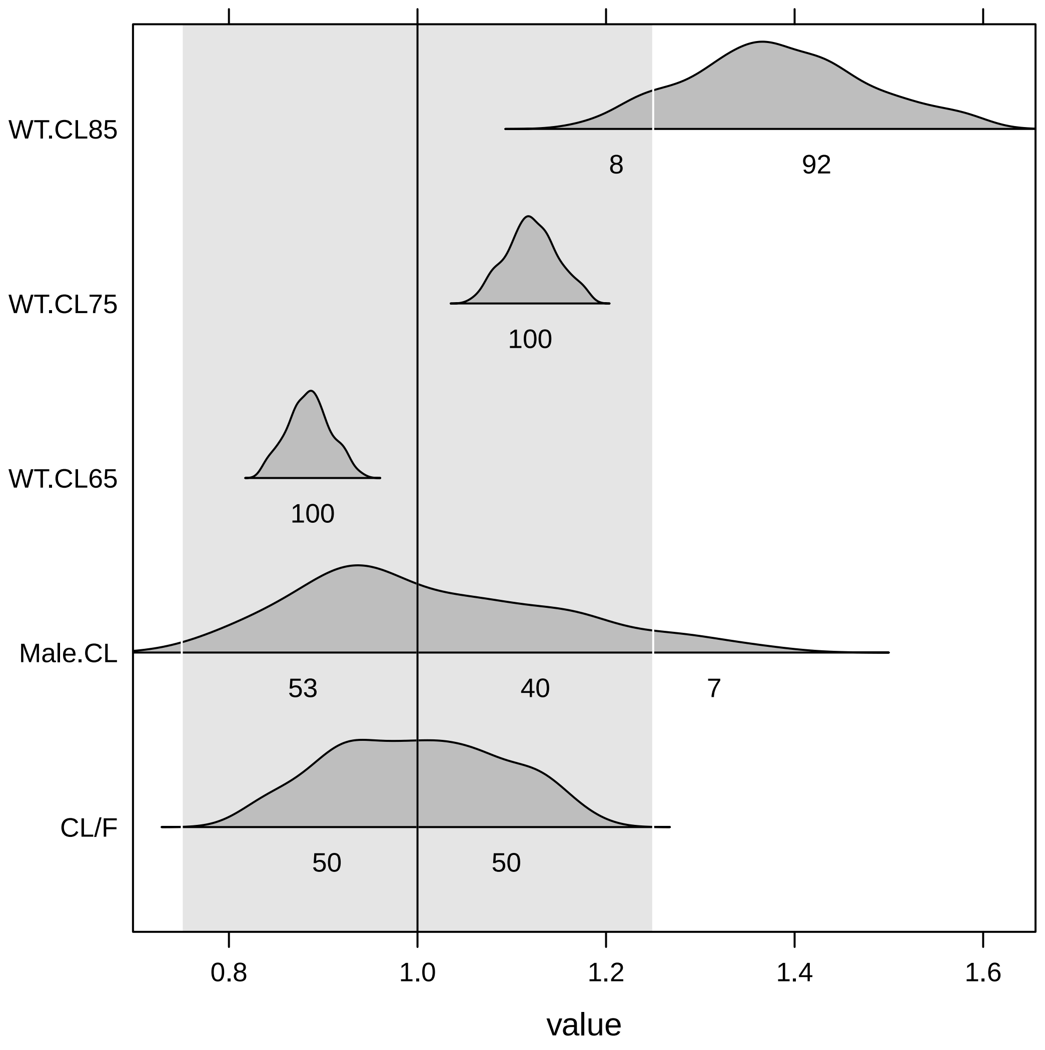
Listing 72:

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

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

Listing 73:

```
> 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 74:

```
> library(Hmisc)
> tab <- partab(1005,'../nonmem',tool='nm7',as=c('label','latex','model','estimate
    ','unit','prse'))
> 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$root <- signif(sqrt(as.numeric(tab$estimate)),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
```

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=44.3)	23.1	(0.115,0.26)
$\Omega^{2.2}Vc/F$		0.129 (%CV=35.9)	30.4	(0.0623,0.181)
$\Omega^{3.3}Ka$		0.107 (%CV=32.7)	25.2	(0.0638,0.157)
$\sigma^{1.1}prop$		0.0671 (%CV=25.9)	11.4	(0.055,0.0796)