Simulation of Clinical Trials Workshop Supplementary Material

Russell Reeve and N. Seth Berry

Code Segments Example 1—Bioequivalence Example with Sample Size Re-Estimation in SAS

```
%let NumIter = 1000;
%let TargetN = 240;
%let MaxN = 300;

data phase1;
input CV  DOrate  InterimPoint;
do iter = 1 to &NumIter;
      do patient=1 to &MaxN;
            day = floor((patient-1)/6) + 1;
            trt = mod(patient-1, 3) + 1;
            if patient<=&TargetN then PlannedGroup=1; else PlannedGroup=0;
            if patient<=&TargetN*InterimPoint then ForInterim=1; else ForInterim=0;
            logCmax = normal(0)*CV;
            if uniform(0) <= DOrate then DO=1; else DO=0;
            if trt=1 then logCmax = logCmax + 0.05;  ** trt=2 and 3 are the reference products **;
            output;
      end;
end;
cards;
0.33   0    0.333
0.33  0.25  0.333
Etc.
;
proc sql;
      create table BlindedInterimResults as
            select CV, DOrate, InterimPoint, iter, N(DO) as NsoFar, mean(DO) as DORateObs,
                        std(logCmax) as CVhat, max(lastPK)+&PKanalysisTime+&AssayTime as
StartAdditionalCohort
            from phase1
            where ForInterim=1
            group by CV, DOrate, InterimPoint, iter;
quit;
data Predicted;
      set BlindedInterimResults;
      NumEvaluable = (1-DORateObs)*&Cohort*&NumCohort/3;
      if NumEvaluable < 60 then MaxN=10; else MaxN=8;

      ***  Calculate Sample Size   ***;
```

```
      CVpt1 = 0.25;     CVpt2 = 0.275;    CVpt3 = 0.3;
      SS1 = 36;   SS2 = 43;   SS3 = 50;
      array CVpt {*} CVpt1-CVpt3;
      array SS {*} SS1-SS3;

      if CVhat <= CVpt1 then SSarm = SS1;
      else
            do i=2 to 12;
                  if CVpt{i-1} < CVhat <= CVpt{i} then SSarm = SS{i};
            end;
      Recruit = SSarm/(1-DORateObs);  ** Per arm sample size **;
      Recruit = 3*Recruit;             ** Sample size total   **;
      Recruit = ceil(max(NsoFar, Recruit));  ** Keep what we already have **;

      keep CV  DOrate  InterimPoint  iter MaxN SSarm Recruit StartAdditionalCohort NsoFar;
run;
proc sort data=phase1;
      by CV  DOrate  InterimPoint iter;
run;
data FinalData;
      merge phase1 Predicted AdditionalCohortsDates MaxPlannedGroup;
      by CV  DOrate  InterimPoint iter;
      if patient <= max(NsoFar, Recruit);
run;
proc sort data=FinalData;
      by CV  DOrate  InterimPoint iter;
run;

ods listing close;
ods output Estimates=estimates;
proc mixed data=FinalData(where=(DO=0));
      by CV  DOrate  InterimPoint iter;
      class trt;
      model logCmax = trt;
      estimate 'TrtDiff 1 vs 2' trt 1 -1 0 / cl alpha=0.1;
      estimate 'TrtDiff 1 vs 3' trt 1 0 -1 / cl alpha=0.1;
      estimate 'TrtDiff 2 vs 3' trt 0 1 -1 / cl alpha=0.1;
run;
ods listing;
data CIinLimits;
      set estimates;
```

```
      if log(0.8) <= lower and upper <= log(1.25) then InLimits=1;
      else InLimits=0;
run;
proc sql;
      create table TrialResults as
            select CV, DOrate, InterimPoint, iter, max(InLimits) as AnyIn, min(InLimits) as AllIn
            from CIinLimits
            group by CV, DOrate, InterimPoint, iter;

quit;
```

Code Segments Example 2—Time Response and Dose Response Model in R

Selected code segments. Not entire program.

Create responses for each subject

```
subject.data <- function(dose=300, dose.effect)
{
    ##### Generates data based on the linear model from the Phase I study
    #####
    ##### Inputs:
    #####      dose.effect = 0 (no effect), 1 (Phase I model), 2 (strong, 15 characters)
    #####
    ##### Outputs:
    #####      data.frame with columns time, val
    #####     Time points are 4, 8, 12, 16, 24 weeks


    ######################################################################
    ### From lm() output on Phase I data
    ##
    ##Coefficients:
    ##                          Estimate Std. Error t value Pr(>|t|)
    ##(Intercept)                xxxxxx   <In actual program had the  output>
    ##week:Dose                  xxxxxxx
    ##week:as.factor(volume)1    xxxxxx
    ##
    ######################################################################

    weeks <- c(4, 8, 12, 16, 24)
    n <- length(weeks)

    eta <- rnorm(1, 0, 10)
    epsilon <- rnorm(n, 0, 10)
```

```
    if(dose==0)
    {
            val <- 0          - xxxxxx*dose*dose.effect + eta + epsilon
    }
    else
    {
            val <- 0          + eta + epsilon
    }

    result <- data.frame(week=weeks, val=val)
    result
}
```

Run Replicated Trials with Parameters Provided

Number of replicates; will want to start with just a few for debugging, and then set to a large number (e.g., 10,000) to compute results

```
design <- function(iter=1, sample.size, dose.effect, alpha=0.05, design)
{
      run.trial <- function(sample.size, dose.effect, design)
      {
            des2 <- design[design$status == 1,]
            DesignPts <- dim(des2)[1]
            NumRep <- trunc( sample.size/DesignPts ) + 1
            doses <- rep(design$dose1, sample.size)[1:sample.size]

            data.out <- data.frame(
               dose1=doses,
               Response=rep(0,length(doses)),
               dose=as.factor(doses
            )

            for(i in 1:dim(data.out)[1])
            {
               subj <- subject.data(dose=data.out[i,"dose1"], dose.effect)
               row <- subj$week==24
```

Simulate trial with patients randomized to various doses

```
            data.out[i,"VAS"] <- subj[row,3]
         }
         trial.lm <<- lm(VAS~dose:volume-1, data=data.out)
         pvalues <- summary(trial.lm)$coefficients[,4]
         pvalues[summary(trial.lm)$coefficients[,1]>0] <- 1
         pvalues
}

found.best <- function(trial.out, alpha, dose.effect, volume.effect, design)
{
      DV <- function(labels)
      {
               N <- length(labels)
               doses <- rep(0, N)
               vols <- rep(0, N)
               DVsplit <- strsplit(labels, ":")
               for(i in 1:N)
               {
                  doses[i] <- as.numeric(strsplit(DVsplit[[i]][1], "dose")[[1]][2])
               }
               data.frame(dose=doses)
      }

      DVoptions <- DV(names(trial.out))
      if(dose.effect==0)
      {
              val <- min(trial.out)>alpha
      }
      else
      {
               #### Where is maximum effect ####
               max.effect <- trial.out == min(trial.out)

               #### Where is the true maximum effect? ####
               max.dose <- max(DVoptions["dose"])
               is.best <- DVoptions$dose==max.dose
```

We need to answer the question: Did we find the best dose?

```
                    val <- any(max.effect & is.best)
            }
            val
      }

      ###### Start computations #######
      K <- sum(design$status)
      test.results <- matrix(rep(NA, (K+2)*iter),  iter, K+2)

      for(i in 1:iter)
      {
            trial.out <- run.trial(sample.size, dose.effect, design)
            test.results[i, 1:K] <- trial.out
            test.results[i, K+1] <- min(trial.out)
            test.results[i, K+2] <- found.best(trial.out, alpha, dose.effect, design)
      }

      ##### Summarize trial results #####
      power <- rep(0, K+2)
      Names <- c(names(trial.out), "overall", "best.dose")
      names(power) <- Names
      for(i in 1:(K+1))
      {
            power[i] <- mean(ifelse(test.results[,i]<alpha,1,0))
      }
      power[K+2] <- mean(test.results[,K+2])
      power
}
```

Run the trials repeatedly, so we can summarize operating characteristics

Calculate power

Run Replicated Trials with Parameters Provided

```
sim <- function(iter=1, dose.cond=c(0, 1, 2, 0, 1, 2), alpha=0.05)
{
      ##### Set everything up
      fixed.power <- function(output)
      {
```

```
        output[(length(output)-2):length(output)]
    }

    K <- length(dose.cond)
    power.curve <- data.frame(P560=rep(0,K), overall=rep(0, K),
                    best.dose=rep(0, K))

    ##### Vary the parameters, and call the simulation function
    for(i in 1:K)
    {
        fixed.output <- design(iter, sample.size=180,
                        dose.effect=dose.cond[i],
                        alpha=0.05, make.design2()
        )
        power.curve[i,] <- fixed.power(fixed.output)
    }
    cbind(data.frame(n=rep(iter, K), dose=dose.cond), power.curve)
}
```