PharmaSUG 2024 – Paper ST-251

Dealing with Missing Data: Practical Implementation in SAS® and R

Isabella Y. Wang, Lauren George, Jin Xie, Eli Lilly and Company, Indianapolis, IN

ABSTRACT

Missing data is a prevalent challenge encountered in Phase III clinical trials. The appropriate handling of missing data becomes crucial and depends on the specific estimand under consideration. In this paper, we will delve into realistic examples of four popular missing data imputation methods and accompany with relevant SAS® and R code. This paper aims to equip clinical trial professionals with practical tools to address missing data effectively and bridge the gap between theoretical understanding and actual application in handling missing data.

INTRODUCTION

Most randomized controlled Phase III clinical trials have some missing data. To minimize the impact of missing data, it is crucial that missing data are addressed appropriately during analysis. Imputation is one main strategy for handling missing data. Generally, imputation methods can be single or multiple imputation. This paper will mainly focus on four imputation methods including Modified Baseline Observation Carried Forward, Non-responder Imputation, Mixed-effects Model for Repeated Measures, and Markov Chain Monte Carlo Multiple Imputation. Data for this paper was generated from simulation R code (APPENDIX I). In each imputation section, only key programming information will be provided. Complete SAS and R code can be found in APPENDIX II & III respectively. All imputation methods were performed using SAS software version 9.4 and RStudio.

IMPUTATION METHODS

Modified Baseline Observation Carried Forward (mBOCF)

For patients discontinuing investigational product (IP) due to treatment-related reasons such as adverse events (AEs) or lack of efficacy, the baseline observation for the endpoint will be carried forward to the corresponding visit for all missing observations after the patient discontinued study treatment. For patients discontinuing IP for any other reason, the last non-missing postbaseline observation before the discontinuation will be carried forward to the corresponding visit(s) for all missing observations after the patient discontinued. For all patients with sporadically missing observations prior to discontinuation, the last non-missing observation before the missing observation will be carried forward to the corresponding visit. Randomized patients without at least one postbaseline observation will not be included for evaluation except for patients discontinuing study treatment due to an AE.

SAS Code

```
/* The last non-missing postbaseline values before discontinuation will be carried
forward for any other discontinuation reason. */
data mbocf_imp;
merge mbocf_obs(in=a) dummy1(in=b);
      by STUDYID SUBJID USUBJID PARAMN PARAM PARAMCD AVISITN;
      length DTYPE $10. AVALC_ $200.;
      retain AVAL_ AVALC_;
       /*Impute last non-missing postbaseline visit*/
       if AVAL ne .  then do;AVAL_=AVAL;AVALC_=AVALC;end;
       if first.PARAMCD and AVAL=. then do;AVAL_=.;AVALC_="";end;
       if not a and b ;
       DTYPE="mBOCF";
```

```
      keep SITEID TRT01P TRT01PN STUDYID SUBJID USUBJID PARAM PARAMN PARAMCD AVISIT
      AVISITN SEX REGION DTYPE AVAL_ AVALC_ DSAEDFL DSAEVIS ISUBJCAT APERIOD APERIODC;
       rename AVAL_=AVAL AVALC_=AVALC;
run;

/* Impute the baseline value for the visits after the discontinue visit if
discontinued due to AE or death. */
data mbocf_imp2;
merge mbocf_imp(in=a) base;
       by STUDYID SUBJID USUBJID PARAMN PARAM PARAMCD;
       if a;
       BASEC=BASEC_;
       if DSAEDFL="Y" then do; /*DSAEDFL = disc. due to AE or death*/
       if AVISITN>DSAEVIS /*DSAEVIS = disc. Visit due to AE or death*/ or AVAL=.
              or AVISITN=DSAEVIS and (AVAL=. or DTYPE="mBOCF")
              then do; AVAL=BASE; AVALC=BASEC; end;
       end;
       if nmiss(AVAL,BASE)=0 then CHG=AVAL-BASE;
       if AVAL ne .;
run;
```

R Code

```
# Impute for the patients with baseline and at least one postbaseline or patients
discontinued from AE and with non-missing baseline.
mbocf_imp <- sim_data %>% filter(PARAMCD == 'NUMVAR') %>% filter(AVISITN>=2) %>%
       mutate(BASEC = as.character(BASEC)) %>%
       full_join(dummy2, by = c("USUBJID", "AVISIT", "AVISITN", "APERIOD", "APERIODC",
                     "STUDYID", "SITEID", "SEX", "REGION", "SUBJID", "DSAEDFL",
                     "DSAEVIS","TRT01P", "TRT01PN", "PARAMN", "PARAM", "PARAMCD",
                     "BASE", "BASEC", "ISUBJCAT")) %>%
       arrange(USUBJID, AVISITN) %>%
       # Impute the baseline value for the visits after the discontinue visit if
       discontinued due to AE or death.
       mutate(DTYPE = case_when(is.na(AVAL) ~ 'mBOCF', TRUE ~ ''),
              AVAL = case_when(is.na(AVAL) & DSAEDFL  == 'Y' & AVISITN >= DSAEVIS
                                   ~BASE, TRUE ~ AVAL)) %>%
                                   # DSAEDFL = discontinuation due to AE or death
                                   # DSAEVIS = discontinuation visit due to AE or death
       group_by(USUBJID) %>%
       tidyr::fill(AVAL, .direction = "down") %>%
       ungroup() %>%
       # For any other reason, the last non-missing postbaseline values before
       discontinuation will be carried forward.
       mutate(AVALC = case_when(is.na(AVALC) ~ as.character(AVAL), TRUE ~ AVALC),
              BASEC = case_when(is.na(BASEC) ~ as.character(BASE), TRUE ~ BASEC),
              CHG = case_when(!is.na(BASE) & !is.na(AVAL) & is.na(CHG) & is.na(ABLFL) ~
                     AVAL-BASE, TRUE ~ CHG))

# Set records together.
mbocf <- rbind(base, mbocf_imp) %>%
       arrange(USUBJID, PARAMN, AVISITN, DTYPE) %>%
       mutate(BASEC = as.character(BASEC),
              ABLFL = case_when(is.na(ABLFL) ~ '', TRUE ~ ABLFL),
              ANLOBSFL = case_when(is.na(ANLOBSFL) ~ '', TRUE ~ ANLOBSFL))
```

Non-responder Imputation (NRI)

NRI method can be used to assess analysis of categorical efficacy and health outcome variables. Patients will be considered a non-responder for the NRI analysis if they do not meet the clinical response criteria or

have missing clinical response data at the timepoint of interest. Randomized patients without at least one post-baseline observation will also be defined as non-responder. NRI method can be justified based on the composite strategy (ICH E9R1) for handling intercurrent events. In this strategy patients are defined as responders only if they meet the clinical requirements for response at the predefined time and they remain on the assigned study treatment. Failing either criterion by definition makes them a non-responder.

SAS Code

```
/* Impute missing visits as non-responder. */
proc sql noprint;
	create table nri_imp as
	select b.*,
		"NRI" as DTYPE length=10,
		0 as AVAL, "N" as AVALC
	from dummy1 as b
	left join nri_obs as a
		on a.USUBJID=b.USUBJID and a.PARAM=b.PARAM and a.PARAMCD=b.PARAMCD and
			a.AVISITN=b.AVISITN
	where a.AVAL=.;
quit;
```

R Code

```
# Impute missing visits as non-responder.
nri_imp <- sim_data %>% filter(PARAMCD == 'CATVAR') %>%
	right_join(dummy2, by = c("USUBJID", "AVISIT", "AVISITN", "APERIOD",
		"APERIODC", "STUDYID", "SITEID", "SEX", "REGION", "SUBJID", "DSAEDFL",
		"DSAEVIS","TRT01P", "TRT01PN", "PARAMN", "PARAM", "PARAMCD")) %>%
	mutate(DTYPE = case_when(is.na(AVAL) ~ 'NRI', TRUE ~ ''),
		ANLOBSFL = case_when(is.na(AVAL) ~ '', TRUE ~ ANLOBSFL),
		ABLFL = case_when(is.na(AVAL) ~ '', TRUE ~ ABLFL),
		AVAL = case_when(is.na(AVAL) ~ 0, TRUE ~ AVAL),
		AVALC = case_when(is.na(AVALC) ~ 'N', TRUE ~ AVALC))
```

Mixed-effects Model for Repeated Measures (MMRM)

For continuous variables, MMRM with missing at random (MAR) assumption can be applied for handling missing data. When using this analysis, both the missingness of data and the correlation of the repeated measurements are considered. The MMRM method may be used both under a treatment policy strategy (ICH E9R1) and under a hypothetical strategy for handling intercurrent events. The hypothetical strategy treats patient data after certain intercurrent events as missing, while the treatment policy strategy uses all available data. Therefore, the missing at random assumption may be justified as consistent with the treatment policy strategy in intent.

SAS Code

```
/* Impute missing visits using proc mixed. */
proc mixed data=mmrm_obs;
	class USUBJID AVISITN TRT01P REGION SEX;
	model CHG=BASE AVISITN TRT01P REGION SEX BASE*AVISITN TRT01P*AVISITN/
		HTYPE=3 DDFM=kr;
	repeated AVISITN/subject=USUBJID type=ar(1);
	lsmeans TRT01P*AVISITN/diff cl;
	ods output lsmeans = anl_lsmean diffs = anl_lsdiff tests3 = anl_effects;
run;
```

R Code

```
# Select analysis dataset.
mmrm.dat <- sim_data %>% filter(PARAMCD == 'NUMVAR') %>%
       filter(AVISITN<=10 & AVISITN>2 & ANLOBSFL=='Y') %>%
       mutate(BASEC = as.character(BASEC),
              TRT01P = factor(TRT01P),
              AVISITN = factor(AVISITN),
              SEX = factor(SEX),
              REGION = factor(REGION))

# Run MMRM function and construct respective statistics.
fit.ls <- mmrm_tryfit(data = mmrm.dat, fixed = "CHG ~ BASE + SEX + REGION + TRT01P +
                           AVISITN + TRT01P * AVISITN", random = "AVISITN | USUBJID")
emmeans.res <- emmeans(fit.ls$model, ~TRT01P | AVISITN, weights = "proportional")
diff.res <- pairs(emmeans.res, reverse = TRUE)
cidif.res <- confint(diff.res)
```

Markov Chain Monte Carlo Multiple Imputation (MCMC-MI)

MCMC-MI can be used to handle both categorical and continuous data. When applying MCMC-MI, the imputation will be conducted within each treatment group independently, so the pattern of missing observations in one treatment group cannot influence missing value estimations in another. In SAS, PROC MI with the MCMC option will be used to conduct the MCMC-MI method, and in R the function 'MISS' is applied. The imputation model will include the relevant baseline and postbaseline observations. In this example, 25 datasets with imputations will be calculated for each imputation process. Each complete dataset will be analyzed with the specified analysis.

SAS Code

```
/* Impute MCMC with all 4 parts subjects for treatment arm. */
proc mi data = indt3 out = outdt_2 seed = 353985587 nimpute = 25 minmaxiter=500;
       mcmc chain = single;
       by PARAMN PARAMCD PARAM;
       var V2-V10;
       where TRT01P='Treatment';
run;

/* Impute MCMC with all 4 parts subjects for placebo arm. */
proc mi data = indt3 out = outdt_1 seed = 1828572477 nimpute = 25  minmaxiter=500;
       mcmc chain = single;
       by PARAMN PARAMCD PARAM;
       var V2-V10;
       where TRT01P='Placebo';
run;
```

R Code

```
# Use MISS() function to impute treatment and placebo arms separately with MCMC.
MISS() function will run a MCMC analysis similar to SAS "proc mi" code, but it
operates under slightly different assumptions, potentially giving different results.
fit.trt <- MISS(indt2_trt_mat, Iterations = (N-1)*N.iter+N.burnin+1, Algorithm = "GS",
       verbose = FALSE) # Impute treatment arm with MISS().

for (i in 1:N){
       indt2_trt_mat_ <- indt2_trt_mat
       indt2_trt_mat_[is.na(indt2_trt_mat)] <- fit.trt$Imp[,(i-1)*N.iter+N.burnin+1]
       indt2_trt[,4:12] <- indt2_trt_mat_
       indt2_trt$AGRPID <- i
```

```
        if (i==1) {
                indt2_trt_imp <- indt2_trt
        } else {
        indt2_trt_imp <- rbind(indt2_trt_imp, indt2_trt)
        }
}

fit.pbo <- MISS(indt2_pbo_mat, Iterations = (N-1)*N.iter+N.burnin+1, Algorithm = "GS",
                verbose = FALSE) # Impute placebo arm with MISS().

for (i in 1:N){
        indt2_pbo_mat_ <- indt2_pbo_mat
        indt2_pbo_mat_[is.na(indt2_pbo_mat)] <- fit.pbo$Imp[,(i-1)*N.iter+N.burnin+1]
        indt2_pbo[,4:12] <- indt2_pbo_mat_
        indt2_pbo$AGRPID <- i

        if (i==1) {
        indt2_pbo_imp <- indt2_pbo
        } else {
        indt2_pbo_imp <- rbind(indt2_pbo_imp, indt2_pbo)
        }
}
```

CONCLUSION

In conclusion, addressing missing data in clinical studies is crucial. Imputation serves as a key strategy in managing this common challenge. This paper explored four prominent imputation methods complete with SAS and R code, thereby enabling statistical programmers to apply these methods seamlessly in their research endeavors. Through this comprehensive analysis, the aim is to contribute to the enhancement of data accuracy and analytical robustness in clinical research.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:

Isabella Y. Wang
wang_yali@lilly.com

Lauren George
lauren.george@lilly.com

Jin Xie
xie_jin_jx@lilly.com

APPENDIX I

Simulated Data R Code

```
library(tidyverse)
set.seed(1)

# Simulate unique subject ID.
N <- 5000

df.subj <- expand.grid(SITEID_ = 4001:4050,
                       SUBJID_ = 1:100,
                       STUDYID = 'J2T-AA-DEMO') %>%
       mutate(SUBJID = sprintf('%1.4d-%2.3d',SITEID_, SUBJID_),
              USUBJID = paste0(STUDYID,'-',SUBJID),
              SITEID = sprintf('%1.4d',SITEID_)) %>%
       select(-SITEID_, -SUBJID_)

# Simulate treatment group.
df.trt <- df.subj %>%
       mutate(TRT01PN_ = runif(N),
              TRT01PN = case_when(TRT01PN_ < 0.5 ~ 1, TRUE ~ 2),
              TRT01P = case_when(TRT01PN == 1 ~ 'Placebo',
              TRT01PN == 2 ~ 'Treatment')) %>%select(-TRT01PN_)
df.trt %>% count(TRT01P)

# Simulate the continuous and categorical parameters and visits.
df.visit <- expand_grid(df.trt, AVISITN = 1:10) %>%
       mutate(AVISIT = case_when(AVISITN == 1 ~ 'Screening',
                                 AVISITN == 2 ~ 'Week 0',
                                 AVISITN == 3 ~ 'Week 2',
                                 AVISITN == 4 ~ 'Week 4',
                                 AVISITN == 5 ~ 'Week 6',
                                 AVISITN == 6 ~ 'Week 8',
                                 AVISITN == 7 ~ 'Week 10',
                                 AVISITN == 8 ~ 'Week 12',
                                 AVISITN == 9 ~ 'Week 14',
                                 AVISITN == 10 ~ 'Week 16'))

set.seed(123)

# Create DSAEDFL to indicate whether the patient discontinued from study treatment due
to AE. Create ISUBJCAT to indicate the subject category: Completed subjects, Subjects
with rescue medication, Subjects due to lack of efficacy, Subjects due to any other
reasons.
df.dsaed <- df.subj %>%
       mutate(DSAEDFL_ = runif(N),
              INTTMFL_ = runif(N),
              DSAEDFL = case_when(DSAEDFL_ <= 0.03 ~ 'Y', TRUE ~ 'N'),
              ISUBJCAT = case_when(DSAEDFL_ <= 0.05 ~ 'Subjects due to any other
                                                      reasons',
                                   DSAEDFL_ <= 0.10 ~ 'Subjects due to lack of
                                                      efficacy',
                                   DSAEDFL_ <= 0.13 ~ 'Subjects with rescue medication',
                                   TRUE ~ 'Completed subjects'),
              MISSVIS = case_when(DSAEDFL_ <= 0.01 ~ 10,
                                  DSAEDFL_ <= 0.02 ~ 9,
                                  DSAEDFL_ <= 0.03 ~ 8,
                                  DSAEDFL_ <= 0.04 ~ 10,
                                  DSAEDFL_ <= 0.05 ~ 8,
```

```
                                DSAEDFL_ <= 0.07 ~ 9,
                                DSAEDFL_ <= 0.09 ~ 8,
                                DSAEDFL_ <= 0.11 ~ 10,
                                DSAEDFL_ <= 0.13 ~ 9),
            INTTMVIS = case_when(INTTMFL_ <=0.05 ~ 4,
                                 INTTMFL_ <= 0.08 ~ 5,
                                 INTTMFL_ <= 0.13 ~ 6,
                                 TRUE ~ 0)
                              )

df.aelong <- df.dsaed %>%
      filter(DSAEDFL_ <= 0.13) %>% select(-DSAEDFL_, -INTTMFL_, -INTTMVIS) %>%
      mutate(MISSVIS8 = case_when(MISSVIS <= 8 ~ 8),
            MISSVIS9 = case_when(MISSVIS <= 9 ~ 9),
            MISSVIS10 = 10) %>%
      pivot_longer(cols = MISSVIS8:MISSVIS10,names_to = "MISSVISN", values_to =
            "AVISITN") %>%
      filter(!is.na(AVISITN)) %>% select(-DSAEDFL, -MISSVISN, -MISSVIS, -ISUBJCAT)

df.visit_ <- df.visit %>%
      left_join(df.aelong, by = c("USUBJID", "SUBJID", "SITEID", "STUDYID",
            "AVISITN")) %>%
      left_join(df.dsaed %>% select(-DSAEDFL_, -INTTMFL_, -INTTMVIS), by =
            c("USUBJID","SUBJID", "SITEID", "STUDYID")) %>%
      mutate(DSAEDFL = case_when(DSAEDFL == 'Y' ~ 'Y', TRUE ~ 'N'),
            DSAEVIS = case_when(DSAEDFL == 'Y' ~ MISSVIS),
            ISUBJCAT = case_when(is.na(ISUBJCAT) ~ 'Completed subjects', TRUE ~
                  ISUBJCAT)) %>%
      left_join(df.dsaed %>% select(USUBJID, INTTMVIS), by = c("USUBJID"))

# Simulate continuous response variables.
df.base <- df.subj %>% mutate(BASE = round(abs(rnorm(N)) * 3 + 15, 3))
df.num <- df.visit_ %>% left_join(df.base, by = c("USUBJID", "SUBJID", "SITEID",
            "STUDYID")) %>%
      mutate(AVAL = case_when(AVISITN == 2 ~ BASE, TRUE ~ BASE +
                  round(runif(nrow(df.visit_)) * (AVISITN-2)*6, 3)),
            AVALC = as.character(AVAL),
            BASEC = as.character(BASE)) %>%
      mutate(PARAM = "Continuous Variable",
            PARAMN = 1,
            PARAMCD = "NUMVAR",
            CHG = case_when(AVISITN >= 3 ~ AVAL - BASE)) %>%
      filter(!(!is.na(MISSVIS) & MISSVIS<=AVISITN)) %>%
      filter(!(INTTMVIS==AVISITN)) %>%
      rename(DCVIS = MISSVIS) %>%
      select(-INTTMVIS)

# Simulate categorical response variables.
df.cat <- df.num %>% rename(AVAL_ = AVAL, AVALC_ = AVALC)  %>%
      mutate(AVAL = case_when(AVAL_ >= 40 ~ 1, TRUE ~ 0 ),
            AVALC = case_when(AVAL == 1 ~ 'Y',
                        AVAL == 0 ~ 'N')) %>%
      mutate(PARAM = "Categorical Variable",
            PARAMN = 2,
            PARAMCD = "CATVAR",
            BASE = NA_real_,
            BASEC = '',
            CHG = NA_real_) %>%
      select(-AVAL_, -AVALC_)

# Combine the continuous and categorical datasets.
df.resp <- rbind(df.num, df.cat)
```

```r
# Create baseline demographics information for modelling.
df.demo <- df.subj %>% mutate(SEX_ = runif(N),
                              SEX = case_when(SEX_ < 0.7 ~ 'M',
                                              TRUE ~ 'F'),
                              REGION_ = runif(N),
                              REGION = case_when(REGION_ <= 0.4 ~ 'US',
                                                 REGION_ <= 0.7 ~ 'Rest of World',
                                                 REGION_ <= 1 ~ 'Europe')) %>%
       select(-SEX_, -REGION_)

# Combine the response variables and the demographics data.
df.final <- df.resp %>% left_join(df.demo, by = c("USUBJID", "SUBJID", "SITEID",
                        "STUDYID")) %>%
       mutate(ANLOBSFL = 'Y',
              ABLFL = case_when(AVISITN == 2 ~ 'Y'),
              APERIOD = case_when(AVISITN >= 3 ~ 1),
              APERIODC = case_when(APERIOD == 1 ~ 'Induction Period')) %>%
       arrange(USUBJID, PARAMN, AVISITN)

# Output the simulated data.
write.csv(df.final, "/drive/data/sim_data.csv", row.names = FALSE, na ='')
```

APPENDIX II

SAS Code

```sas
/* Import data and set formatting */
proc import
       datafile="/drive/data/sim_data.csv"
       out=work.sim_data
       dbms=csv
       replace; GUESSINGROWS=1000;
run;

proc format;
       value vis
              1 = "Screening"      2 = "Week 0"        3 ="Week 2"
              4 = "Week 4"         5 = "Week 6"        6 ="Week 8"
              7 = "Week 10"        8 = "Week 12"       9 ="Week 14"
              10 = "Week 16"       ;

       picture pv (round)
              0 - < 0.001        ="<0.001" (noedit)
              0.001 - < 0.9995   ="9.999"
              0.9995  - 1        =">0.999" (noedit)
              .                  ="   -   " ;
    ;
quit;
```

Modified Baseline Observation Carried Forward (mBOCF)

```
/* Load observed simulation data. */
data mbocf_obs;
set sim_data;
       where AVAL ne .  and PARAMCD in ("NUMVAR") and ANLOBSFL="Y";
       drop DCVIS;
run;

/* Only impute if subject has at least one post baseline record unless discontinuing
due to AE. */
proc sort data=mbocf_obs out=subj_num(keep=STUDYID SUBJID USUBJID SITEID TRT01P
       TRT01PN SEX REGION PARAMN PARAMCD PARAM DSAEDFL DSAEVIS ISUBJCAT) nodupkey;
       by STUDYID SUBJID USUBJID PARAMN;
       where AVISITN>=3 or (AVISITN=2 and DSAEDFL="Y");
run;

/* Generate dummy shell. */
data dummy1;
set subj_num;
       APERIOD=1;
       APERIODC="Induction Period";
       do AVISITN=3 to 10;
              AVISIT=put(AVISITN, vis.);
       output;
       end;
run;

/* If the subject discontinued from treatment for any other reason, the last non-
missing postbaseline values before discontinuation will be carried forward to the
corresponding visit for all missing observations after the patient discontinued. */
data mbocf_imp;
merge mbocf_obs(in=a) dummy1(in=b);
       by STUDYID SUBJID USUBJID PARAMN PARAM PARAMCD AVISITN;
       length DTYPE $10. AVALC_ $200.;
       retain AVAL_ AVALC_;
       /*Impute last non-missing postbaseline visit*/
       if AVAL ne .  then do;AVAL_=AVAL;AVALC_=AVALC;end;
       if first.PARAMCD and AVAL=. then do;AVAL_=.;AVALC_="";end;
       if not a and b ;
       DTYPE="mBOCF";
       keep SITEID TRT01P TRT01PN STUDYID SUBJID USUBJID PARAM PARAMN PARAMCD AVISIT
              AVISITN SEX REGION DTYPE AVAL_ AVALC_ DSAEDFL DSAEVIS ISUBJCAT APERIOD
              APERIODC;
       rename AVAL_=AVAL AVALC_=AVALC;
run;

/* If subject discontinued from treatment due to AE or Death, then impute the baseline
value for the visit after the discontinue visit for mBOCF. */
proc sort data=mbocf_obs out=base(keep=STUDYID SUBJID USUBJID PARAMN PARAM PARAMCD
AVAL AVALC rename=(AVAL=BASE AVALC=BASEC_));
       by STUDYID SUBJID USUBJID PARAMN PARAM PARAMCD;
       where ABLFL="Y";
run;

data mbocf_imp2;
merge mbocf_imp(in=a) base;
       by STUDYID SUBJID USUBJID PARAMN PARAM PARAMCD;
       if a;
       BASEC=BASEC_;
       if DSAEDFL="Y" then do; /*DSAEDFL = disc. due to AE or death*/
       if AVISITN>DSAEVIS /*DSAEVIS = disc. Visit due to AE or death*/ or AVAL=.
              or AVISITN=DSAEVIS and (AVAL=. or DTYPE="mBOCF")
```

```
            then do; AVAL=BASE; AVALC=BASEC; end;
      end;
      if nmiss(AVAL,BASE)=0 then CHG=AVAL-BASE;
      if AVAL ne .;
run;

/* Set imputed and observed records together to create final dataset. */
data mbocf;
set mbocf_imp2 mbocf_obs;
run;
```

Non-responder Imputation (NRI)

```
/* Load observed simulation data. */
data nri_obs;
set sim_data;
      where PARAMCD in ("CATVAR") and ANLOBSFL="Y" and AVAL ne .;
      drop DSAEDFL DSAEVIS DCVIS ISUBJCAT;
run;

/* Select subjects and generate shell dataset in order to impute missing visits as
Non-responders. */
proc sort data=nri_obs out=subj_cat(keep=STUDYID SUBJID USUBJID SITEID TRT01P: SEX
REGION PARAM:) nodupkey;
      by STUDYID SUBJID USUBJID PARAMN;
      where AVISITN>=2;
run;

data dummy1;
set subj_cat;
      APERIOD=1;
      length APERIODC $18. AVISIT $11.;
      APERIODC="Induction Period";
      do AVISITN=3 to 10;
        AVISIT=put(AVISITN, vis.);
      output;
      end;
run;

/* Impute missing visits as Non-responders. */
proc sql noprint;
      create table nri_imp as
      select b.*,
            "NRI" as DTYPE length=10,
            0 as AVAL, "N" as AVALC length=8
      from dummy1 as b
      left join nri_obs as a
            on a.USUBJID=b.USUBJID and a.PARAM=b.PARAM and a.PARAMCD=b.PARAMCD and
                  a.AVISITN=b.AVISITN
      where a.AVAL=.;
quit;

/* Set imputed and observed records together to create final dataset.*/
data nri;
set nri_imp nri_obs;
run;
```

Mixed-effects Model for Repeated Measures (MMRM)

```
/* Load observed simulation data. */
data mmrm_obs;
set sim_data;
       where PARAMCD="NUMVAR" and ANLOBSFL="Y" and APERIOD=1 and CHG ne .;
run;

/* Impute missing using proc mixed. */
ods exclude all;
proc mixed data= mmrm_obs;
       class USUBJID AVISITN TRT01P REGION SEX;
       model CHG=BASE AVISITN TRT01P REGION SEX BASE*AVISITN TRT01P*AVISITN/
              HTYPE=3 DDFM=kr;
       repeated AVISITN/subject=USUBJID type=ar(1);
       lsmeans TRT01P*AVISITN/diff cl;
       ods output lsmeans = anl_lsmean diffs = anl_lsdiff tests3 = anl_effects;
run;
ods exclude none;

/* Format the output statistics. */
data mmrm;
merge anl_lsmean(in=a rename=(Estimate=est_lsm Probt=pval_lsm StdErr=SD_lsm)
                     drop=UPPER LOWER)
      anl_lsdiff(in=b where=(AVISITN=_AVISITN) drop=TRT01P rename=(Estimate=est_dif
                     Probt=pval_dif StdErr=SD_dif _TRT01P=TRT01P));

       by AVISITN TRT01P;
       if a then do;
       length LSM PVALTRT $20.;
       LSM=put(est_lsm,7.3)||" ("||put(SD_lsm,7.3)||")";
       if .<pval_lsm<0.001 then PVALTRT="<0.00001";
       else if pval_lsm^=. then PVALTRT=put(pval_lsm, 7.5);
       end;

       if b then do;
       length LSMDIF CIDIF PVALDIF $20.;
       LSMDIF=put(-est_dif,7.3)||" ("||put(sd_dif,7.3)||")";
       CIDIF="("||put(-Upper,7.3)||", "||put(-Lower,7.3)||")";
       if .<pval_dif<0.00001 then PVALDIF="<0.00001";
       else if pval_dif^=. then PVALDIF=put(pval_dif, 7.5);
       end;

       keep AVISITN TRT01P LSM PVALTRT CIDIF LSMDIF PVALDIF;
run;
```

Markov Chain Monte Carlo Multiple Imputation(MCMC-MI)

```
/* Read observed data from source and identify which part each subject belongs to. */
data indt1;
       set sim_data;
       where PARAMCD="NUMVAR" and (ABLFL="Y" OR (2<AVISITN<=10 and ANLOBSFL='Y' ));
         if ISUBJCAT='Subjects with rescue medication' then PART=1;
         else if ISUBJCAT='Subjects due to lack of efficacy'  then PART=2;
         else if ISUBJCAT='Subjects due to any other reasons'  then PART=3;
         else if ISUBJCAT='Completed subjects'  then PART=4;
       drop DSAEDFL DSAEVIS;
run;
```

```
/* Set aval to missing before imputation for records after intercurrent events. */
proc sort data = indt1 nodupkey out = dummy (keep = STUDYID SITEID SUBJID USUBJID
TRT01P: PARAM: PART BASE: SEX REGION DCVIS ISUBJCAT);
  by USUBJID;
run;

data dummy2;
       set dummy;
       do AVISITN = 2 to 10;
       AVISIT=put(AVISITN, vis.);
       output;
       end;
run;

data indt2;
       merge indt1(in=b) dummy2(in=a);
       by USUBJID AVISITN;
       if a;
       if b then OC=1;
       if (AVISITN>DCVIS>.Z) then do;
              if PART in (1,2)  then do; AVAL=BASE; BSTPFL=1; end;
              if PART=3  then AVAL=.;
              OC=.;
       end;
       drop PART;
run;

proc sort data = indt2;
       by PARAM: TRT01P: STUDYID SITEID SUBJID USUBJID BASE: SEX REGION ISUBJCAT
       DCVIS;
run;

proc transpose data= indt2 out=indt3 prefix=V;
       by  PARAM: TRT01P: STUDYID SITEID SUBJID USUBJID BASE: SEX REGION ISUBJCAT
       DCVIS;
       var AVAL;
       id AVISITN;
run;

/* Impute MCMC with all 4 parts subjects for placebo arm. */
ods exclude all;
proc mi data = indt3 out = outdt_1 seed = 1828572477 nimpute = 25  minmaxiter=500;
       mcmc chain = single;
       by PARAMN PARAMCD PARAM;
       var V2-V10;
       where TRT01P='Placebo';
run;
ods exclude none;

/* Impute MCMC with all 4 parts subjects for treatment arm. */
ods exclude all;
proc mi data = indt3 out = outdt_2 seed = 353985587 nimpute = 25 minmaxiter=500;
       mcmc chain = single;
       by PARAMN PARAMCD PARAM;
       var V2-V10;
       where TRT01P='Treatment';
run;
ods exclude none;

/* Set all imputation records together. */
proc sql;
       create table outdt as
       select * from outdt_1
```

```
      union select * from outdt_2
      order by PARAMN, PARAMCD, PARAM, STUDYID, SITEID, SUBJID, USUBJID,
            _IMPUTATION_, TRT01PN, TRT01P, BASE, BASEC, SEX, REGION, ISUBJCAT, DCVIS;
quit;

proc transpose data=outdt out=outdt_t;
      by  PARAM: STUDYID SITEID SUBJID USUBJID _IMPUTATION_ TRT01P: BASE: SEX REGION
      ISUBJCAT DCVIS;
      var V2-V10 ;
run;

data outdt_t1;
      set outdt_t;
      AVISITN=input(compress(_NAME_,'V'),best.);
      where not missing(aval);
run;

proc sort data=outdt_t1 out=base_deriv(keep=USUBJID _IMPUTATION_ AVAL);
      by USUBJID _IMPUTATION_;
      where AVISITN=2;
run;

proc sql;
      create table outdt_t2 as
      select a.*,
            input(compress(a._NAME_,'V'),best.) as AVISITN,
            b.OC,
            b.BSTPFL,
            'Y' as ANLMCFL,
            a._IMPUTATION_ as AGRPID,
            'Primary (Hybrid)' as ESTIMAND,
            case when b.BSTPFL=1 then 'Imputed Baseline'
                  when b.OC NE 1 then 'MCMC-MI'
                  else '' end as DTYPE,
            case when input(compress(a._NAME_,'V'),best.) NE . then
                  put(input(compress(a._NAME_,'V'),best.), vis.)
                  end as AVISIT
      from outdt_t as a
      left join indt2 as b
      on a.USUBJID = b.USUBJID and input(compress(a._NAME_,'V'),best.) = b.AVISITN
      order by USUBJID, _IMPUTATION_;
quit;

data outdt_t2a;
      merge outdt_t2(in=a drop=BASE) base_deriv(rename=AVAL=BASE);
      by USUBJID _IMPUTATION_;
      if a;
run;

proc sql;
      create table outdt_t3 as
      select * from indt1
      outer union corr select * from outdt_t2a
      order by USUBJID;
quit;

proc sql;
      create table outdt_t4 as
      select a.*, b.ORI_BASE as BASE, b.ORI_BASE
      from outdt_t3(drop=BASE) as a
      left join (
            select USUBJID, PARAMCD, AVAL as ORI_BASE
            from indt1
```

```
            where ABLFL='Y'
      ) as b
      on a.USUBJID = b.USUBJID and a.PARAMCD = b.PARAMCD
      order by USUBJID;
quit;

/* Create final dataset */
data mcmc;
      set outdt_t4;
      if  ANLMCFL='Y'  AND AVISITN>2 then do;
            if nmiss(AVAL, ORI_BASE)=0 then CHG=AVAL- ORI_BASE;
            APERIOD=1;
            APERIODC="Induction Period";
      end;
      if AVAL ne . then AVALC=strip(put(AVAL,BEST.));
      drop _IMPUTATION_ _NAME_ OC BSTPFL ORI_BASE;
run;
```

APPENDIX III

R Code

```
library(tidyverse)
library(stringr)
library(mmrm)
library(emmeans)
library(LaplacesDemon)

datapath <- '/drive/data'
sim_data <- read.csv(file.path(datapath,"sim_data.csv"))
```

Modified Baseline Observation Carried Forward (mBOCF)

```
subj <- sim_data %>% filter(PARAMCD == 'NUMVAR') %>%
      select(STUDYID, SUBJID, USUBJID, SITEID, TRT01P, TRT01PN, SEX, REGION, PARAMN,
            PARAMCD, PARAM, DSAEDFL, DSAEVIS, BASE, BASEC, ISUBJCAT) %>%
      mutate(BASEC = as.character(BASEC)) %>%
      distinct(STUDYID, SUBJID, USUBJID, PARAMN, .keep_all = TRUE)

# Categorize baseline and postbaseline subjects.
base <- sim_data %>% filter(ABLFL == 'Y', PARAMCD == 'NUMVAR') %>%
            select(USUBJID)
postbase <- sim_data %>% filter(PARAMCD == 'NUMVAR', AVISITN >= 3) %>%
            distinct(USUBJID)

# Create a vector of patients with baseline and at least one postbaseline, along with
dummy shell.
eff.subj <- base %>% inner_join(postbase, by = "USUBJID") %>%
      pull(USUBJID)

dummy.vis <- sim_data %>% filter(PARAMCD == 'NUMVAR') %>%
      distinct(AVISIT, AVISITN, APERIOD, APERIODC) %>%
      filter(AVISITN>=2)
```

```
dummy.subj <- subj %>% filter((DSAEDFL=='Y' & USUBJID %in% pull(base,USUBJID))|
             USUBJID %in% eff.subj)
dummy1 <- expand_grid(dummy.vis, USUBJID = unique(dummy.subj$USUBJID))
dummy2 <- dummy1 %>% left_join(subj, by = c("USUBJID"))

# Impute for the patients with baseline and at least one postbaseline or patients
discontinued from AE and with non-missing baseline.
mbocf_imp <- sim_data %>% filter(PARAMCD == 'NUMVAR') %>%
       filter(AVISITN>=2) %>%
       mutate(BASEC = as.character(BASEC)) %>%
       full_join(dummy2, by = c("USUBJID", "AVISIT", "AVISITN", "APERIOD", "APERIODC",
                    "STUDYID", "SITEID", "SEX", "REGION", "SUBJID", "DSAEDFL",
                    "DSAEVIS","TRT01P", "TRT01PN", "PARAMN", "PARAM", "PARAMCD",
                    "BASE", "BASEC", "ISUBJCAT")) %>%
       arrange(USUBJID, AVISITN) %>%
       # Impute the baseline value for the visits after the discontinue visit if
       discontinued due to AE or death.
       mutate(DTYPE = case_when(is.na(AVAL) ~ 'mBOCF', TRUE ~ ''),
              AVAL = case_when(is.na(AVAL) & DSAEDFL  == 'Y' & AVISITN >= DSAEVIS
                                ~BASE, TRUE ~ AVAL)) %>%
       group_by(USUBJID) %>%
       tidyr::fill(AVAL, .direction = "down") %>%
       ungroup() %>%
       # For any other reason, the last non-missing postbaseline values before
       discontinuation will be carried forward.
       mutate(AVALC = case_when(is.na(AVALC) ~ as.character(AVAL), TRUE ~ AVALC),
              BASEC = case_when(is.na(BASEC) ~ as.character(BASE), TRUE ~ BASEC),
              CHG = case_when(!is.na(BASE) & !is.na(AVAL) & is.na(CHG) & is.na(ABLFL) ~
                    AVAL-BASE, TRUE ~ CHG))

# Set records together.
base <- sim_data %>% filter(PARAMCD == 'NUMVAR') %>% filter(AVISITN<2) %>%
       mutate(DTYPE = '')
mbocf <- rbind(base, mbocf_imp) %>%
       arrange(USUBJID, PARAMN, AVISITN, DTYPE) %>%
       mutate(BASEC = as.character(BASEC),
              ABLFL = case_when(is.na(ABLFL) ~ '', TRUE ~ ABLFL),
              ANLOBSFL = case_when(is.na(ANLOBSFL) ~ '', TRUE ~ ANLOBSFL))
```

Non-responder Imputation (NRI)

```
# Generate dummy shell dataset in order to impute missing visits.
subj <- sim_data %>% filter(PARAMCD == 'CATVAR') %>%
       select(STUDYID, SUBJID, USUBJID, SITEID, TRT01P, TRT01PN, SEX, REGION, PARAMN,
              PARAMCD, PARAM, DSAEDFL, DSAEVIS) %>%
       distinct(STUDYID, SUBJID, USUBJID, PARAMN, .keep_all = TRUE)

dummy.vis <- sim_data %>% distinct(AVISIT, AVISITN) %>%
       filter(AVISITN >= 3)

dummy1 <- expand_grid(dummy.vis, APERIOD = 1, APERIODC = "Induction Period", USUBJID =
             unique(sim_data$USUBJID))

dummy2 <- dummy1 %>% left_join(subj, by = c("USUBJID"))

# Impute missing visits as non-responders.
nri_imp <- sim_data %>% filter(PARAMCD == 'CATVAR') %>%
       right_join(dummy2, by = c("USUBJID", "AVISIT", "AVISITN", "APERIOD",
                           "APERIODC", "STUDYID", "SITEID", "SEX", "REGION", "SUBJID",
                           "DSAEDFL", "DSAEVIS","TRT01P", "TRT01PN", "PARAMN",
                           "PARAM", "PARAMCD")) %>%
```

```
      mutate(DTYPE = case_when(is.na(AVAL) ~ 'NRI', TRUE ~ ''),
          ANLOBSFL = case_when(is.na(AVAL) ~ '', TRUE ~ ANLOBSFL),
          ABLFL = case_when(is.na(AVAL) ~ '', TRUE ~ ABLFL),
          AVAL = case_when(is.na(AVAL) ~ 0, TRUE ~ AVAL),
          AVALC = case_when(is.na(AVALC) ~ 'N', TRUE ~ AVALC))

# Set imputed and observed records together to create final dataset.
base <- sim_data %>% filter(PARAMCD == 'CATVAR', AVISITN<=2) %>%
      mutate(DTYPE='')

nri <- rbind(base, nri_imp) %>%
      arrange(USUBJID, PARAMN, AVISITN, DTYPE) %>%
      select(-DSAEDFL, -DSAEVIS) %>%
      mutate(BASEC = as.character(BASEC))
```

Mixed-effects Model for Repeated Measures (MMRM)

```
# This function returns a list containing the MMRM model fitted (mmrm object) and the
covariance structure used (character object).
mmrm_tryfit <- function(data, fixed, random) {

# Try the covariance structure in the order of us/toeph/ar1h/csh/toep/ar1/cs until the
convergence is met.
df_covlist <- data.frame(covtype = c("us", "toeph", "ar1h", "csh", "toep", "ar1",
                                     "cs"),
                         covlabel = c("Unstructured",
                                      "Heterogeneous Toeplitz",
                                      "Heterogeneous Autoregressive",
                                      "Heterogeneous Compound Symmetry",
                                      "Homogeneous Toeplitz",
                                      "Homogeneous First Order Autoregressive",
                                      "Homogeneous Compound Symmetry"))

    for (i in 1:nrow(df_covlist)){

        # Construct formula based on user input.
        model_formula_char <- paste0(fixed, " + ", df_covlist[i, 1], "(", random, ")")
        model_formula <- model_formula_char %>% as.formula()

        Fit the mmrm model.
        fit_mmrm <- mmrm(
            formula = model_formula,
            data = data,
            reml = TRUE,
            method = "Kenward-Roger",
            optimizer = "nlminb"
        )

        # If the model converges, break the loop.
        if (component(fit_mmrm, "convergence") == 0) {
         cat(paste("Model with", df_covlist[i, 2], "covariance structure converged and
             outputted."),
                paste("Formula used:", model_formula_char),
                sep = "\n")
          break

        # If the model does not converge, try the next structure.
        } else {
       cat(paste("Model with", df_covlist[i, 2], "covariance structure did not
       converge."),
                paste("Formula used:", model_formula_char),
```

```
                sep = "\n")

            if (df_covlist[i, 1] != "cs") {
                cat("Trying the next covariance structure ...\n")
            } else {
           stop("None of the required covariance structures can lead to a converged
      MMRM model.")
            }
        }
    }
    output_list <- list(model = fit_mmrm, covlabel = df_covlist[i, 2])
    return(output_list)
}

# Select analysis dataset.
mmrm.dat <- sim_data %>% filter(PARAMCD == 'NUMVAR') %>%
      filter(AVISITN<=10 & AVISITN>2 & ANLOBSFL=='Y') %>%
      mutate(BASEC = as.character(BASEC),
             TRT01P = factor(TRT01P),
             AVISITN = factor(AVISITN),
             SEX = factor(SEX),
             REGION = factor(REGION))

# Run MMRM function and construct respective statistics.
fit.ls <- mmrm_tryfit(data = mmrm.dat,
             fixed = "CHG ~ BASE + SEX + REGION + TRT01P + AVISITN + TRT01P*AVISITN",
             random = "AVISITN | USUBJID")
emmeans.res <- emmeans(fit.ls$model, ~TRT01P | AVISITN, weights = "proportional")
test.res <- test(emmeans.res)
diff.res <- pairs(emmeans.res, reverse = TRUE)
cidif.res <- confint(diff.res)

# Format statistics.
emmeans.df <- data.frame(test.res) %>%
      mutate(LSM = sprintf("%1.3f(%2.3f)", emmean, SE),
             PVALTRT = case_when(p.value < 0.00001 ~ '<0.00001', TRUE ~
                                 sprintf("%1.5f",p.value))) %>%
      select(AVISITN, TRT01P, LSM, PVALTRT)

diff.df <- data.frame(diff.res) %>%
      mutate(LSMDIF = sprintf("%1.3f(%2.3f)", estimate, SE),
             PVALDIF = case_when(p.value < 0.00001 ~ '<0.00001', TRUE ~
                       sprintf("%1.5f",p.value)),
             TRT01P = 'Treatment') %>%
      select(AVISITN, TRT01P, LSMDIF, PVALDIF)
cidif.df <- data.frame(cidif.res) %>%
      mutate(CIDIF = sprintf("(%1.3f,%2.3f)", lower.CL, upper.CL), TRT01P =
             'Treatment') %>%
      select(AVISITN, TRT01P, CIDIF)

mmrm.res <- emmeans.df %>% left_join(diff.df, by = c("TRT01P", "AVISITN")) %>%
      left_join(cidif.df, by = c("TRT01P", "AVISITN"))
```

Markov Chain Monte Carlo Multiple Imputation(MCMC-MI)

```
# Read observed data from source.
indt0 <- sim_data %>% filter(PARAMCD == 'NUMVAR') %>%
      filter((AVISITN<=10 & AVISITN>2 & ANLOBSFL=='Y')|ABLFL == 'Y') %>%
      mutate(BASEC = as.character(BASEC))
```

```
# Generate dummy dataset in order to impute missing visits.
subj <- sim_data %>% filter(PARAMCD == 'NUMVAR') %>%
       select(STUDYID, SUBJID, USUBJID, SITEID, TRT01P, TRT01PN, SEX, REGION, PARAMN,
              PARAMCD, PARAM, BASE, BASEC, ISUBJCAT, DCVIS) %>%
       mutate(BASEC = as.character(BASEC)) %>%
       distinct(STUDYID, SUBJID, USUBJID, PARAMN, .keep_all = TRUE)

dummy.vis <- sim_data %>% filter(PARAMCD == 'NUMVAR') %>%
       distinct(AVISIT, AVISITN, APERIOD, APERIODC) %>%
       filter(AVISITN>=2)
dummy1 <- expand_grid(dummy.vis, USUBJID = unique(subj$USUBJID))
dummy2 <- dummy1 %>% left_join(subj, by = c("USUBJID"))

# Decide which part each subject belongs to, and set aval to missing before imputation
for records after intercurrent events. For subjects who discontinued due to lack of
efficacy or use of rescue medication, impute as baseline.
indt0_ <- indt0 %>% mutate(OC = 1)

indt1 <- dummy2 %>% left_join(indt0_, by = c("USUBJID", "AVISIT", "AVISITN",
                     "APERIOD", "APERIODC", "STUDYID", "SITEID", "SEX", "REGION",
                     "SUBJID", "TRT01P", "TRT01PN", "PARAMN", "PARAM", "PARAMCD",
                     "BASE", "BASEC", "ISUBJCAT", "DCVIS")) %>%
       mutate(AVAL = case_when(ISUBJCAT %in% c("Subjects due to lack of efficacy",
                     "Subjects with rescue medication") & AVISITN > DCVIS ~ BASE, TRUE
                     ~ AVAL),
              AVALC = case_when(ISUBJCAT %in% c("Subjects due to lack of efficacy",
                     "Subjects with rescue medication") & AVISITN > DCVIS ~ BASEC, TRUE
                     ~ AVALC),
              BSTPFL = case_when(ISUBJCAT %in% c("Subjects due to lack of efficacy",
                     "Subjects with rescue medication") & AVISITN > DCVIS ~ 1,
                     TRUE ~ 0)) %>%
       arrange(USUBJID, AVISITN)

# Fill in the other missingness with MCMC multiple imputation and build 2 separate
chains by treatment.
indt2 <- indt1 %>% select(USUBJID, AVISITN, AVAL, TRT01P, TRT01PN) %>%
       pivot_wider(names_from = AVISITN, names_prefix = "V", values_from = AVAL)

indt2_trt <- indt2 %>% filter(TRT01P == "Treatment")
indt2_pbo <- indt2 %>% filter(TRT01P == "Placebo")

indt2_trt_mat <- indt2_trt %>% select(-USUBJID, -TRT01P, -TRT01PN) %>% as.matrix()
indt2_pbo_mat <- indt2_pbo %>% select(-USUBJID, -TRT01P, -TRT01PN) %>% as.matrix()

setTimeLimit(cpu = Inf, elapsed = Inf, transient = FALSE) # Avoid program pausing due
                                                            to CPU time limit.
set.seed(12345) # Set random seed to control the results.
N <- 25 # Number of imputation.
N.burnin <- 200 # Number of burn-ins.
N.iter <- 100 # Number of iterations between imputations.

# Use MISS() function to impute treatment and placebo arms separately with MCMC.
MISS() function will run a MCMC analysis similar to SAS "proc mi" code, but it
operates under slightly different assumptions, potentially giving different results.
fit.trt <- MISS(indt2_trt_mat, Iterations = (N-1)*N.iter+N.burnin+1, Algorithm = "GS",
              verbose = FALSE) # Impute treatment arm with MISS().

for (i in 1:N){
  indt2_trt_mat_ <- indt2_trt_mat
  indt2_trt_mat_[is.na(indt2_trt_mat)] <- fit.trt$Imp[,(i-1)*N.iter+N.burnin+1]
  indt2_trt[,4:12] <- indt2_trt_mat_
  indt2_trt$AGRPID <- i
```

```
  if (i==1) {
    indt2_trt_imp <- indt2_trt
  } else {
     indt2_trt_imp <- rbind(indt2_trt_imp, indt2_trt)
    }
}

indt2_trt_imp_t <- indt2_trt_imp %>% pivot_longer(names_to = "AVISITN", values_to =
       "AVAL", cols = V2:V10, names_prefix = "V")

setTimeLimit(cpu = Inf, elapsed = Inf, transient = FALSE) # Avoid program pausing due
                                                            to CPU time limit.
set.seed(12345) # Set random seed to control the results.

fit.pbo <- MISS(indt2_pbo_mat, Iterations = (N-1)*N.iter+N.burnin+1, Algorithm = "GS",
             verbose = FALSE) # Impute placebo arm with MISS().

for (i in 1:N){
  indt2_pbo_mat_ <- indt2_pbo_mat
  indt2_pbo_mat_[is.na(indt2_pbo_mat)] <- fit.pbo$Imp[,(i-1)*N.iter+N.burnin+1]
  indt2_pbo[,4:12] <- indt2_pbo_mat_
  indt2_pbo$AGRPID <- i

  if (i==1) {
    indt2_pbo_imp <- indt2_pbo
  } else {
    indt2_pbo_imp <- rbind(indt2_pbo_imp, indt2_pbo)
  }
}

indt2_pbo_imp_t <- indt2_pbo_imp %>% pivot_longer(names_to = "AVISITN", values_to =
       "AVAL", cols = V2:V10, names_prefix = "V")

indt2_imp <- rbind(indt2_trt_imp_t, indt2_pbo_imp_t)

indt3 <- indt2_imp %>% left_join(subj, by = c("USUBJID", "TRT01P", "TRT01PN")) %>%
       mutate(AVALC = as.character(round(AVAL,3)),
              AVISITN = as.numeric(AVISITN),
              ESTIMAND = "Induction Primary (Hybrid)",
              ANLMCFL = 'Y',
              ANLOBSFL = '',
              CHG = case_when(AVISITN > 2 ~ AVAL - BASE, TRUE ~ NA_real_)) %>%
       left_join(dummy.vis, by = "AVISITN")

# Transform the observed data.
indt4 <- indt0 %>%
       mutate(ESTIMAND = '', ANLMCFL = '', AGRPID = NA_integer_, DTYPE = '') %>%
       select(-DSAEDFL, -DSAEVIS, -ABLFL)

indt5 <- indt1 %>% select(USUBJID, AVISITN, OC, BSTPFL)

# Create DTYPE and combine the observed and imputed data.
indt6<- indt3 %>% left_join(indt5, by = c("USUBJID", "AVISITN")) %>%
       mutate(DTYPE = case_when(BSTPFL == 1 ~ 'Imputed Baseline',
              is.na(OC) ~ 'MCMC-MI', TRUE ~ '')) %>%
       select(-OC, -BSTPFL)

final <- rbind(indt6,indt4) %>%
       arrange(USUBJID, AVISITN, !is.na(AGRPID)) %>%
       mutate(ABLFL = case_when(AVISITN <= 2 & DTYPE == '' & BASEC != '' & ESTIMAND ==
                     '' ~ 'Y', TRUE ~ ''))
```