Supplemental material to
"A comparison of multiple imputation methods for incomplete longitudinal binary data"

1 Sample SAS codes

Our analyses in the simulation study and the application were performed by using `SAS` software. In particular, we used `MI` procedure for imputing the missing data. We below provide sample `SAS` codes for implimenting the imputation in the six MI methods described in Section 2. Table S1 shows a part of input dataset which is represented as `indata` in the sample codes. The input dataset `indata` contains nine variables. `SUBJID` is a character variable representing a subject identification number. `GROUPN` and `GROUP` are a numeric and a character variable respectively, representing group indicators. `_1W`, `_2W`, `_3W`, `_4W`, `_5W` and `_6W` are numeric variables taking binary values 0 or 1 (e.g. 0 for non-responder and 1 for responder).

Table S1. Input dataset `indata`.

SUBJID	GROUPN	GROUP	_1W	_2W	_3W	_4W	_5W	_6W
S1001	1	Active	0	1	1	1	1	0
S1004	1	Active	0	1	0	0	1	1
S1005	1	Active	0	1	1	.	.	.
S1006	1	Active	1	1	1	1	.	.
S1008	1	Active	0	0	0	1	1	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
S1007	0	Placebo	1	0	0	1	1	0
S1012	0	Placebo	0	.	.	.	.	.
S1015	0	Placebo	0	0	0	1	0	.
S1024	0	Placebo	0	0	0	1	1	1
S1025	0	Placebo	0	0	1	.	.	.
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

1.1 Monotone method

The imputations using the monotone method with the logistic regression and the discriminant function are performed, and their results are stored in output datasets `MONOL` and `MONOD` respectively.

```
proc mi data=indata nimpute=40 out=MONOL;
  class GROUP _1W _2W _3W _4W _5W _6W;
  monotone logistic(_1W _2W _3W _4W _5W _6W);
  var GROUP _1W _2W _3W _4W _5W _6W;
run;

proc mi data=indata nimpute=40 out=MONOD;
  class GROUP _1W _2W _3W _4W _5W _6W;
  monotone discrim(_1W _2W _3W _4W _5W _6W/CLASSEFFECTS=INCLUDE);
  var GROUP _1W _2W _3W _4W _5W _6W;
run;
```

1.2 Full conditional specification (FCS)

The imputations using the FCS with the logistic regression and the discriminant function are performed, and their results are stored in output datasets `FCSL` and `FCSD` respectively.

```
proc mi data=indata nimpute=40 out=FCSL;
  class GROUP _1W _2W _3W _4W _5W _6W;
  fcs logistic(GROUP _1W _2W _3W _4W _5W _6W);
  var GROUP _1W _2W _3W _4W _5W _6W;
run;

proc mi data=indata nimpute=40 out=FCSD;
  class GROUP _1W _2W _3W _4W _5W _6W;
  fcs discrim(GROUP _1W _2W _3W _4W _5W _6W/CLASSEFFECTS=INCLUDE);
  var GROUP _1W _2W _3W _4W _5W _6W;
run;
```

1.3 Markov chain Monte Carlo (MCMC)

The imputations using the MCMC with the coin flipping and the adaptive rounding are performed, and their results are stored in output datasets `MCMCC` and `MCMCA` respectively.

```
proc mi data=indata nimpute=40 out=WKMC1;
  mcmc chain=single initial=em;
  var GROUPN _1W _2W _3W _4W _5W _6W;
run;

data WKMC2;
  set WKMC1;
  if _1W<0 then I_1W=0; else if _1W>1 then I_1W=1; else I_1W=_1W;
  if _2W<0 then I_2W=0; else if _2W>1 then I_2W=1; else I_2W=_2W;
  if _3W<0 then I_3W=0; else if _3W>1 then I_3W=1; else I_3W=_3W;
  if _4W<0 then I_4W=0; else if _4W>1 then I_4W=1; else I_4W=_4W;
```

```
  if _5W<0 then I_5W=0; else if _5W>1 then I_5W=1; else I_5W=_5W;
  if _6W<0 then I_6W=0; else if _6W>1 then I_6W=1; else I_6W=_6W;
  drop _1W _2W _3W _4W _5W _6W;
run;

data MCMCC;
  set WKMC2;
  _1W=rand('bernoulli',I_1W);
  _2W=rand('bernoulli',I_2W);
  _3W=rand('bernoulli',I_3W);
  _4W=rand('bernoulli',I_4W);
  _5W=rand('bernoulli',I_5W);
  _6W=rand('bernoulli',I_6W);
  drop I_:;
run;

proc sort data=WKMC2 out=WKMCA1;
  by _Imputation_ GROUPN;
run;

proc means data=WKMCA1 mean;
  by _Imputation_ GROUPN;
  var I_1W I_2W I_3W I_4W I_5W I_6W;
  ods output Summary=WKMCA2;
run;

proc transpose data=WKMCA2 out=WKMCA3;
  by _Imputation_ GROUPN;
  var I_1W_Mean I_2W_Mean I_3W_Mean I_4W_Mean I_5W_Mean I_6W_Mean;
run;

data WKMCA4;
  set WKMCA3;
  COL2=COL1-quantile('NORMAL',COL1)*sqrt(COL1*(1-COL1));
  drop _LABEL_;
run;

proc transpose data=WKMCA4 out=WKMCA5(drop=_NAME_);
  by _Imputation_ GROUPN;
  id _NAME_;
  var COL2;
run;

data MCMCA;
  merge WKMCA1 WKMCA5;
  by _Imputation_ GROUPN;
  if I_1W<I_1W_Mean then _1W=0; else _1W=1;
  if I_2W<I_2W_Mean then _2W=0; else _2W=1;
```

```
  if I_3W<I_3W_Mean then _3W=0; else _3W=1;
  if I_4W<I_4W_Mean then _4W=0; else _4W=1;
  if I_5W<I_5W_Mean then _5W=0; else _5W=1;
  if I_6W<I_6W_Mean then _6W=0; else _6W=1;
  keep _Imputation_ SUBJID GROUP: _:;
run;
```