2014 ■ vol. 10 ■ no. 2

DOI: 10.20982/tqmp.10.2.p153

Handling missing data in cluster randomized trials: A demonstration of multiple imputation with PAN through SAS

Jiangxiu Zhou [a], Lauren E. Connell [a], John W. Graham [a]

[a] Department of Biobehavioral Health, The Pennsylvania State University

Abstract ■ The purpose of this study is to demonstrate a way of dealing with missing data in clustered randomized trials by doing multiple imputation (MI) with the PAN package in R through SAS. The procedure for doing MI with PAN through SAS is demonstrated in detail in order for researchers to be able to use this procedure with their own data. An illustration of the technique with empirical data was also included. In this illustration the PAN results were compared with pairwise deletion and three types of MI: (1) Normal Model (NM)-MI ignoring the cluster structure; (2) NM-MI with dummy-coded cluster variables (fixed cluster structure); and (3) a hybrid NM-MI which imputes half the time ignoring the cluster structure, and the other half including the dummy-coded cluster variables. The empirical analysis showed that using PAN and the other strategies produced comparable parameter estimates. However, the dummy-coded MI overestimated the intraclass correlation, whereas MI ignoring the cluster structure and the hybrid MI underestimated the intraclass correlation. When compared with PAN, the p-value and standard error for the treatment effect were higher with dummy-coded MI, and lower with MI ignoring the cluster structure, the hybrid MI approach, and pairwise deletion. Previous studies have shown that NM-MI is not appropriate for handling missing data in clustered randomized trials. This approach, in addition to the pairwise deletion approach, leads to a biased intraclass correlation and faulty statistical conclusions. Imputation in clustered randomized trials should be performed with PAN. We have demonstrated an easy way for using PAN through SAS.

Keywords ■ PAN, missing data, cluster randomized trials, multiple imputation, SAS

jgraham@psu.edu

Introduction

Multilevel, or clustered study designs, are research designs where some independent variables are measured at the cluster level and the dependent variable is measured at the individual level. The cluster randomized trial (CRT) is a type of multilevel study where randomization of the intervention occurs at the cluster level and is conducted with intact clusters of subjects (e.g., see Murray, 1998). CRTs are commonly used in prevention research, especially in studies with children and adolescents where treatments are typically randomized at the school level.

The missing data issue is common in prevention research. Normal model multiple imputation (MI) is often used for dealing with missing data in CRTs with two main approaches for handling the cluster structure: (1) ignoring the cluster structure during imputation or (2) including dummy codes to represent the clusters in the MI model. However, neither approach is appropriate. Ignoring the cluster structure during imputation is the same as imputing under a model where all cluster means are the same; this biases the cluster means toward the grand mean and attenuates differences between clusters. Taljaard et al. (2008) has suggested that ignoring the cluster structure during imputation leads to underestimated between-cluster variance, especially when the true intra-class correlation (ICC) is large (e.g., $> .01$). Including cluster membership dummy codes in the MI model was supposed to solve the problem by allowing different cluster means. However with this approach, imputed values are biased toward the cluster means, thus inflating differences between clusters. Andridge (2011) has suggested that this approach, which represents a fixed effect for cluster, leads to overestimation of the between-cluster variance and ICC, especially when the true ICC is small (e.g., $< .01$). Both approaches have implications for statistical conclusions: ignoring the cluster structure in MI is

associated with inflated type I error, and assuming a fixed cluster effect in MI is associated with inflated type II error (decreases power). The correct MI model is the one that assumes a random effect for the cluster variable.

Many prevention researchers use SAS for multilevel analysis as it has some convenient procedures for multilevel models (e.g., proc mixed, proc genmod etc.). However, procedures for performing MI with random cluster effects are not currently available in SAS. The PAN program (Schafer, 2001; Schafer & Yucel, 2002), which was designed for multiple imputation of multivariate panel or clustered data and is currently only available with R software, is the only tool at present that can incorporate random cluster effects in the MI model.

PAN Available Now, Through SAS

Due to the utility and frequency with which SAS is used by prevention researchers, we introduce a procedure for running the PAN program through SAS. The procedure makes use of a macro for calling R programs from SAS (Xin, 2012) and is easy to use. Instructions for doing so are illustrated step by step in the next section. By replacing the relevant variables in the example with one's own data, one can perform the PAN imputation in SAS.

Method

Empirical Data Used for Illustration

The data for this example come from one cohort of the Adolescent Alcohol Prevention Trial (AAPT; Hansen & Graham, 1991), which aimed to change substance use behavior in adolescents from 12 schools. Four conditions (three interventions and one control) were randomized to the 12 schools when the students were in the 7th grade. For this illustration, we dummy coded program membership so that each dummy variable represented one kind of program versus control (prog1 = program 1 vs. control; prog2 = program 2 vs. control; prog3 = program 3 vs. control). Variables used in our illustration were summarized in Table 1. We included the 3014 participants with complete data for the following six variables: prog1, prog2, prog3, gender, cola1 and cola2. Outcome variable was lifetime cigarette smoking behavior (lifesmk9) in the 9th grade. Other variables included in the imputation were smoking behavior and alcohol use measured in the 7th, 8th, and 9th grade (except for "lifesmk9", which was the outcome variable), and friend's and peer's substance use, measured in the 7th and 8th grade. Nine of these variables had less than 1% missingness, two had around 37% missingness, four had around 22% missingness, two had around 52% missingness, and four had around 45% missingness (see Table 1).

Step-by-Step Illustration of Running PAN Imputation through SAS

Step 1. Install R on your computer. The R software is free and can be downloaded from http://www.r-project.org. Install R on your computer.

Step 2. Install package PAN in R. Open R by double clicking the R icon. You will see the "R Console" window. PAN can be installed in two ways: 1) Copy the following statement into the R Console and hit enter: "install.packages("pan",repo="http://cran.cnr.berekely .edu/")"; or 2) Use the menu "Packages>Install package(s)>pan" and choose the closest CRAN mirror from "CRAN mirror" listings.

To check if PAN was successfully installed, enter "library(pan)" into the Console. If you get the message "Error in library(pan): there is no package called 'pan'", PAN was not successfully installed on your computer. You could try installing the package again using another CRAN mirror. You can find the full CRAN mirror list at http://cran.rproject.org/mirrors.html; choose one close to you to replace "http://cran.cnr.berkeley.edu/" in the statement above. If you do not get an error message, this means that the PAN package was installed successfully and you can proceed to the next step.

Step 3. Get the macro "Proc_R.sas" ready. Download the macro from http://www.jstatsoft.org/v46/c02 and save it to your hard drive. Open the macro in a text editor (e.g., Notepad) and modify line 46, which specifies the R path. The bold underlined text shown below is what needs to be modified. Modify this text to the location of the R package on your computer's hard drive. Save the changes, close the text editor and open SAS. For example,

```
%macro quit(rpath=%str(C:\progra~1\R\R-
2.15.1\bin\R.exe));
```

[note that "progra~1" is DOS shorthand for "program files"; you can also copy/paste the full path from windows explorer navigation bar].

Step 4. Get the R file "diagnosticplots. r" ready. Download the R file from the attachment and save it to your computer.

Table 1 ■ *Variables Used in the Multiple Imputation Model.*

Variables	Label	Percent of missingness
school	Cluster variable	0%
prog1	Program 1	0%
prog2	Program 2	0%
prog3	Program 3	0%
gender	Male/female	0%
cola1	Number of cola drinks past month	0%
cola2	Number of cola drinks past week	0%
frndalc	Number of 3 best friends ever tried alcohol	0.07%
frndsmk	Number of 3 best friends ever tried smoking cigarettes	0.30%
frndmar	Number of 3 best friends ever tried smoking marijuana	0.17%
coffee1	Number of coffee drinks in whole life	0.27%
coffee2	Number of coffee drinks in past month	0.13%
lifesmk7	Number of cigarettes smoked in whole life (grade 7)	0.13%
rcntsmk7	Number of cigarettes smoked in past month (grade 7)	0.13%
lifealc7	Number of alcoholic drinks in whole life (grade 7)	0.53%
rcntalc7	Number of alcoholic drinks in past month (grade 7)	0.23%
peeralc7	Out of every 100 students your age, how many drink alcohol at least once a month (grade 7)	36.79%
peersmk7	Out of every 100 students your age, how many smoke cigarettes at least once a month (grade 7)	36.86%
lifesmk8	Number of cigarettes smoked in whole life (grade 8)	21.83%
rcntsmk8	Number of cigarettes smoked in past month (grade 8)	21.90%
lifealc8	Number of alcoholic drinks in whole life (grade 8)	21.90%
rcntalc8	Number of alcoholic drinks in past month (grade 8)	21.73%
peeralc8	Out of every 100 students your age, how many drink alcohol at least once a month (grade 8)	51.96%
peersmk8	Out of every 100 students your age, how many smoke cigarettes at least once a month (grade 8)	52.32%
lifesmk9	Number of cigarettes smoked in whole life (grade 9)	45.52%
rcntsmk9	Number of cigarettes smoked in past month (grade 9)	45.45%
lifealc9	Number of alcoholic drinks in whole life (grade 9)	45.45%
rcntalc9	Number of alcoholic drinks in past month (grade 9)	45.52%

Step 5.1. Run PAN imputation through SAS: preliminary model specification and diagnostic plots check. The first step of running MI with PAN is to check the diagnostic plots. Interpretation of diagnostic plots for PAN imputation is similar to that for normal model MI. Checking these plots helps to verify that the imputation model is specified appropriately (e.g., enough iterations in the burn-in period and between imputations). In the illustration (Listing 1), we specified a preliminary imputation model, and generated a series of diagnostic plots.

The syntax in Listing 1 includes two parts: preliminary model specification and diagnostic plots generation. Places requiring modifications are shown in bold text. The modifications required for the preliminary model specification (lines 1-34) are:

(1) specify where the SAS macro "Proc_R.sas" downloaded in step 3 was located on your computer by replacing the bold text, "**C:\Users\xxx\Downloads\Proc_R.sas**" in line 3;

(2) specify the dataset to be imputed by replacing the bold text, **b.alc** in lines 4 and 13 with the name of your dataset;

(3) specify a working directory for R by replacing "**C:\temp**" in line 11 with any directory on your computer if it does not exist on your computer. This facilitates output display in the SAS window; you need not pay much attention to anything saved under the directory;

(4) specify the cluster variable in line 15 by replacing the bold text, "**school**" with the name of your cluster variable;

(5) Replace the bold text, "**female","cola1","cola2"** in line 18 with variables that do not have any missing data.

Here in the illustration there are only three variables without missing data but more than three variables can be added. If there is no variable without missing data, delete the bold text in line 18 (e.g., covname=c()). In the imputation process, these variables are used to predict variables with missing data. A random-intercepts, fixed-slopes model is assumed for all variables in most CRTs, so that is the model that is emphasized here (if you are not sure whether or not you want to use a random-intercepts, random-slopes model for analysis and for imputation, we recommend you stay with the random-intercepts, fixed-slopes model). If there are variables assumed to have random slopes (e.g., due to study design or from previous findings), enter these variables first before entering variables with fixed slopes and check the diagnostic plots before proceeding to multiple imputation (as discussed below).

(6) specify the number of variables assumed to have a random slope in step (5) by replacing the bold text, **0**, in line 22, with the number of variables with random slopes;

(7) specify variables with missing data to be included in the imputation model by replacing the bold text in lines 26-28. In the example, we have 21 variables with missing data (of course you can have fewer or more variables here);

(8) Random intercepts in multilevel models are generally estimated with two parameters: a fixed term and a random term. For example, the equation for a two-level random intercept model is $y_{ij} = b_{00} + b_{i0} + e_{ij,}$ where 'i' represents the cluster, 'j' the subject, and where b_{00} is the fixed term and b_{i0} is the random term of the intercept. However, in some situations (as shown later with our example), inclusion of the fixed term in the imputation model might cause convergence problems. The fixed term can be excluded from the multilevel model by replacing the bold **1** in line 31 with a 2. Note that the intercept will still be modeled as a random effect because the random term is always included in the imputation model;

(9) specify the number of iterations for the burn-in period by replacing the bold text, **5000** in line 32 with the desired value. Generally it is recommended to start with 1000 and increase the number of iterations if the diagnostic plots are not stabilized.

(10) Specify the location of file "diagnosticplots.r" downloaded at step 4 by replacing the bold text in line 33.

(11) specify the saving destination of the diagnostic plots by replacing the bold text in line 34. Three types of diagnostic plots (time series plot and autocorrelation plot for simulated values over iterations) are generated: the beta plots, sigma plots, and psi plots, which are stored in three subfolders of the specified location. For more discussion about the diagnostic plots, please see Graham (2012; Chapter 3 and 7).

Troubleshooting

You might get error messages and warnings from the SAS log when running the syntax. If you have a large number of variables included in the imputation model, you might get an error message similar to the one below:

```
ERROR: File is in use,
C:\temp\r_log_1687692869.txt.
```

When this happens, the R log is not displayed in the SAS output window although the imputation may still run correctly, provided that the model is specified correctly. To solve the problem, some minor modifications to the macro Proc_R.sas need to be made:

(1) open the macro in a text editor and go to line 36 and line 91 (refer to Figure 1, Error 1); change the bold text "**datetime()**" in both lines to "abc" (or some other combination of letters as long as they are the same in both lines);

(2) go to line 165 and add the bold statements in the demonstration from Figure 1, Error 1 (part 2) to the macro. Save the macro and try running the imputation model again. If the error message still appears, you can try increasing the number in the last bold statement (e.g., %test(10000000) to %test(20000000)).

You might also get the following warning message when you run the syntax:

```
WARNING: Data too long for column
"R_OUTPUT_LOG"; truncated to 120 characters
to fit.
```

This will not have any impact on the performance of the imputation model except that the R log will not be shown completely in the SAS output window. The warning message can be ignored or you can open the macro Proc_R.sas and add the bold statement shown in Figure 1, Error 2. If the warning message persists, change the 'linesize' from 150 to a larger number.

You might also get error messages from the R log which appears in the SAS output window. An error message of this kind is shown below.

Error 1:

```
...
data _null_;
      call symput('r_code','r_code'||trim(left(scan(put
      (datetime(),best.),1,'.'))));
run;
...
****get current time before run batch R***;
data _null_;
      call symput('nowtime',trim(left(scan(put
      (datetime(),best.),1,'.'))));
run;
%put $$$$&nowtime;
...
```

← Line 36

← Line 91

Error 1 (part 2)

```
...
****get current time after run batch R***;
data _null_;
    call symput('aftertm',trim(left(put(datetime(),datetime19.))));
run;
%put $$$$&aftertm;

%macro test(finish);
   %let i=1;
   %do %while (&i<&finish);
         %let i=%eval(&i+1);
   %end;
%mend test;
%test(10000000);

data rlog;
     infile "&sasdirec\r_log_&nowtime..txt" length=len;
     input var1 $varying8192. len;
run;
...
```

← Use a number larger than 10000000 as needed

Error 2:

```
...
title "******************R OUTPUT**********************";
options linesize=150;

proc print data=rlog(rename=(var1=R_OUTPUT_LOG)) noobs; run;
title;
...
```

Figure 1 ■ Troubleshooting error messages when running the SAS macro.

`Error: XXXXXXXXXXXXXXXXXX. Execution halted.`

This often results from an error when replacing the variables with your own data and variable names. Check the syntax carefully; be sure that variable substitution was done correctly. The error message often gives some hint as to where the error in the syntax occurred. For example, if the error message says "unexpected symbol in yname=c("frndalc,"frndsmk"", this indicates that something is wrong with this line and we can see that the quotation marks were not closed for "frndalc".

Diagnostic Plots

If the imputation model runs without any error messages, you can check the diagnostic plots by

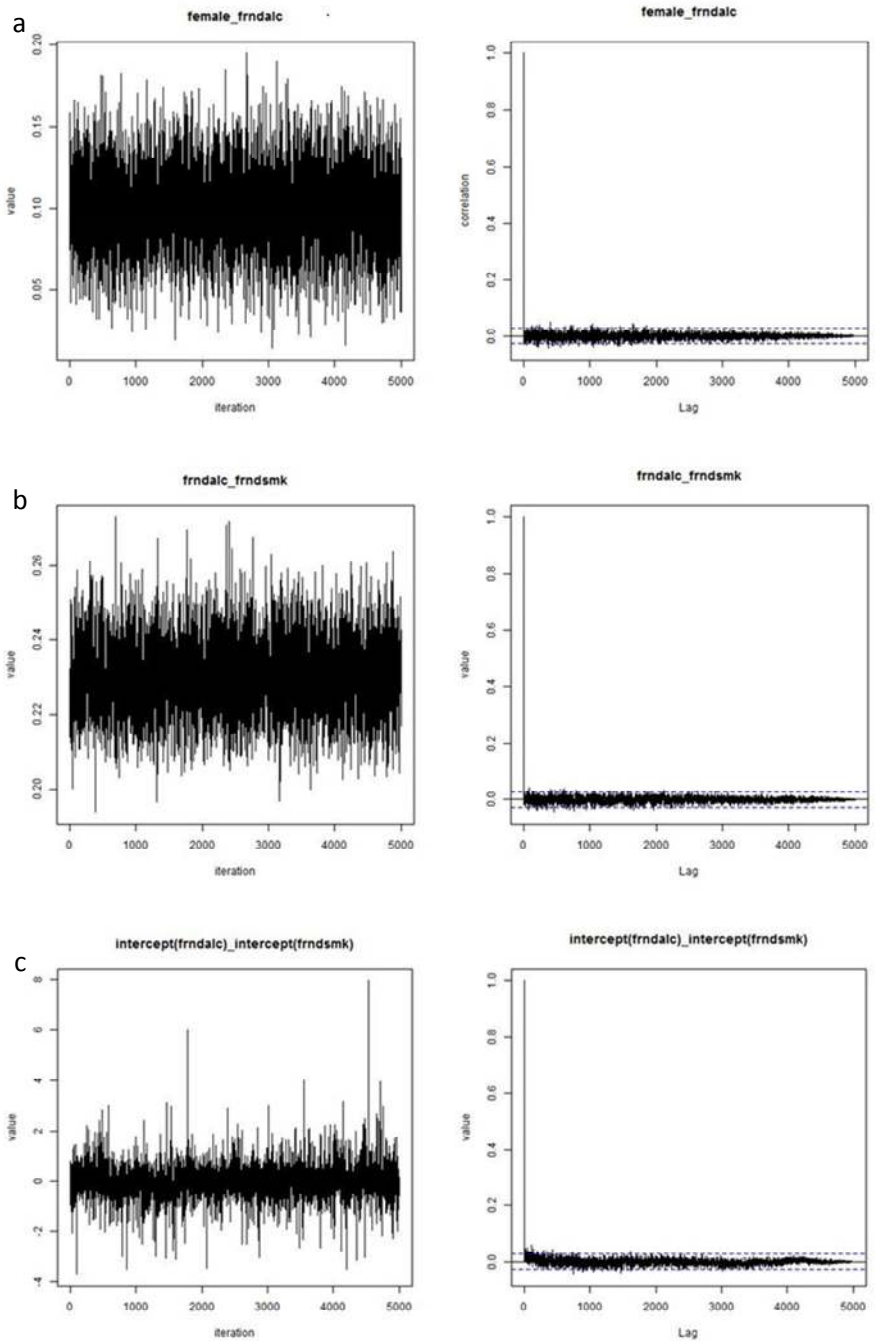

Figure 2 ■ Diagnostic plots for 5,000 burn-in iterations to predict values on the variable 'frndalc'. Panel (a) displays beta plots, Panel (b) displays sigma plots and Panel (c) displays psi plot; The left column contains time series plots and the right column contains autocorrelation plots

opening the folder to which you specified the plots be saved.

PAN produces two kinds of plots (time series and autocorrelation) for three types of parameters (beta, sigma, and psi). Beta parameters are fixed parameter estimates (or fixed term of random parameter estimates) of variables without missing data predicting variables with missing data. The title of the beta plot indicates the parameter. For example, the title of the plots in Figure 2, panel (a) "female_frndalc" indicates that the parameter is the regression coefficient for the variable "female" predicting "frndalc". The beta plots

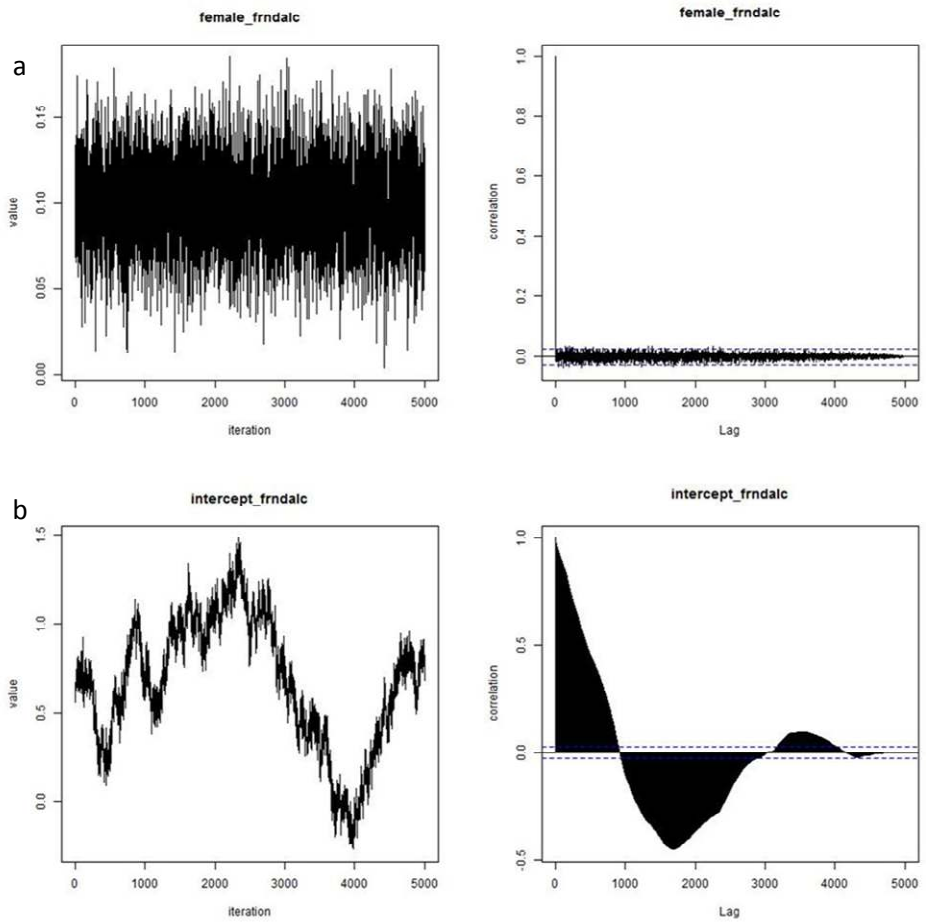

Figure 3 ■ Diagnostic beta plots over 5,000 burn-in iterations for predicting the variable 'frndalc'. Panel (a) displays an acceptable plot and panel (b) displays a clearly pathological plot. The left column contains time series plots and the right column contains autocorrelation plots.

should be checked to confirm that the plots are clearly acceptable. To be considered acceptable, the time series plots should be in a rectangular shape and the autocorrelation plots should fall below the horizontal line that indicates the significance level (for more information about interpreting these plots, see Graham, 2012; chapter 3 and 7). Pathological plots were observed for some of the parameter estimates in our illustration (Figure 3). The time series plots did not stabilize even after 5000 iterations and the corresponding autocorrelation plots indicated high parameter correlations across iterations. Closer examination revealed that the problematic plots were all intercept estimates (plot titles were all in the form of "intercept_xxx"). On some occasions, specifying more iterations in the burn-in period could solve the problem. This was not a solution for our data. Even after 30,000 iterations, the plots were still pathological. Thus we excluded the fixed intercept term (by modifying line 31), which resulted in acceptable beta plots (similar to those shown in Figure 2, panel (a)).

Sigma parameters represent the residual variances and covariances. The title of the sigma plot indicates the parameter. For example in Figure 2, panel (b), "frndalc_frndsmk" indicates that the parameter is the residual covariance between the variables "frndalc" and "frndsmk".

Psi parameters represent the random effects variances and covariances. The title of the psi plot indicates the parameter estimated. For example, the title of plots in Listing 3, panel (c) "intercept(frndalc)_intercept(frndsmk)" indicates that the parameter is the covariance between the random intercepts for the variables "frndalc" and "frndsmk".

Both sigma and psi plots appear to be acceptable after the modification (similar to those shown in panel (b) and (c) of Figure 2).

Step 5.2. Run PAN imputation through SAS: specify sufficient number of iterations between imputations by checking diagnostic plots for one single imputation.

First, look at the autocorrelation plots from the burn-in period. We used 5000 burn-in iterations, but it may be sufficient to use some smaller number (e.g., 1000). Examine all of the autocorrelation plots to determine a conservative number of iterations that are needed between imputations; this is done by identifying the number where it is clear that the autocorrelations have fallen below the significance lines. The number of iterations necessary will vary from study to study. In our case, 1000 iterations was a conservative number of iterations between imputations.

Next, generate the diagnostic plots for a single imputation by following the steps below:

(1) keep lines 1 to 55 as specified in Listing 1 and modify line 31 by replacing 1 with **2** (see previous discussion about diagnostic plots);

(2) ask for the conservative number of iterations you picked by replacing **1000** in line 57 in the syntax shown in Listing 2;

(3) specify the directory where you want the diagnostic plots to be saved by replacing the bold text in line 58. Refer to the previous troubleshooting section if there are any error messages or warnings.

Examine the diagnostic plots (especially the autocorrelation plots) from these (post-burn-in) iterations. With the smaller number of iterations, it will typically be possible to verify with some certainty that the number of iterations chosen (1000 in our case) was sufficient to simulate multiple random draws from the population (e.g., see Graham, 2012). In many cases, this examination of the plots will show that some smaller number of iterations between imputations will be acceptable. Based on our examination of the autocorrelation plots, it was appropriate to conduct multiple imputations with 1000 iterations between imputations (Figure 4).

Step 6. Run PAN imputation through SAS: conducting multiple imputation. The diagnostic plots generated in step 5.2 verified that 1000 iterations between imputations were sufficient to simulate random draws from the population and we can proceed to MI. The syntax for PAN imputation (Listing 3) excludes the section generating diagnostic plots in step 5.1 and adds a new section for multiple imputation. Keep all the modifications you made before with the preliminary model specification (lines 1-57) except for line 4 and make two additional modifications to the syntax in Listing 3:

(1) specify the name of the PAN imputed dataset by replacing the bold text "**b.pan**" in line 4 and line 79 with the name of the resulting dataset including the imputed cases;

(2) specify the number of imputations in line 63 by replacing the bold text "**40**" with the number of imputations you would like. Here we asked for 40 imputations. Refer to the previous troubleshooting section if there are any error messages or warnings.

(3) specify the name of variables that are not included in the imputation but will be used for further analysis by replacing the bold text in line 65.

Step 7. Modifying the imputed dataset for analysis in SAS. Before conducting further analysis in SAS, generate variable "_imputation_" from variable "imputation" in the PAN imputed dataset. Please see the example SAS syntax below:

```
data b.pan; *<-Replace with your imputed
dataset name;

set b.pan;

_imputation_=imputation;

Run;
```

Empirical Illustration of Analysis with Imputed Datasets Using PAN

Method

The imputed dataset from PAN can be used for analysis like any other MI data set. We show an example of fitting a linear model with the "PROC MIXED" procedure using the treatment variables (prog1, prog2, and prog3) to predict lifetime smoking in grade 9 (lifsmk9; example SAS syntax are shown below).

```
proc mixed data=b.pan noclprint covtest;

title 'lifesmk9';
  class school;
  model lifesmk9=prog1 prog2 prog3 lifealc7
lifesmk7/solution ddfm=bw;
  random intercept/sub=school;
  by _imputation_;
```

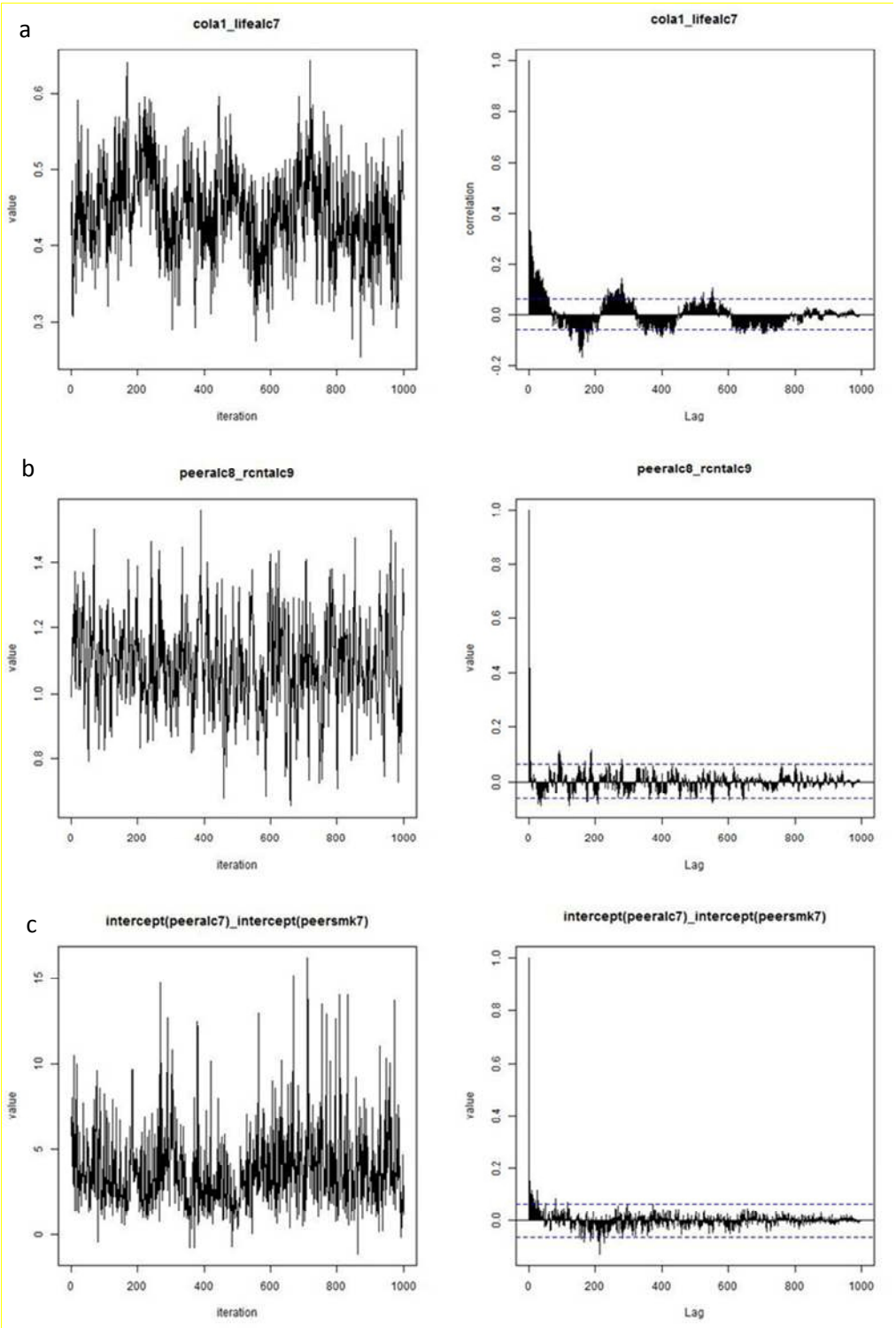

Figure 4 ■ Worst diagnostic plots for 1,000 iterations between imputations. Panel (a) displays beta plots, Panel (b) displays sigma plots and Panel (c) displays psi plots. The left column contains time series plots and the right column contains autocorrelation plots.

```
  ods output solutionf=b.mixparms
covparms=b.mixparmsR;run;
```

Table 2 ■ Parameter estimates of program membership predicting lifetime smoking behavior in grade 9 (lifesmk9).

Program	Imputation methods	Regression Coefficient	SE	t	df	p	ICC*
Prog1	PAN	-0.174	0.172	-1.01	5.54	0.354	0.012
	Dummy-coding	-0.170	0.173	-0.98	5.59	0.366	0.012
	Ignoring cluster	-0.153	0.128	-1.20	5.03	0.285	0.004
	Hybrid	-0.157	0.147	-1.07	5.72	0.328	0.008
	Pairwise Deletion	-0.165	0.151	-1.09	8	0.307	0.007
Prog2	PAN	-0.421	0.176	-2.39	5.53	0.057	
	Dummy-coding	-0.405	0.180	-2.25	5.36	0.071	
	Ignoring cluster	-0.425	0.139	-3.05	4.56	**0.032**	
	Hybrid	-0.423	0.154	-2.74	5.48	**0.037**	
	Pairwise Deletion	-0.428	0.155	-2.76	8	**0.025**	
Prog3	PAN	-0.142	0.168	-0.84	5.92	0.432	
	Dummy-coding cluster	-0.130	0.171	-0.76	5.81	0.476	
	Ignoring cluster	-0.116	0.128	-0.91	5.25	0.403	
	Hybrid	-0.111	0.151	-0.74	5.53	0.491	
	Pairwise Deletion	-0.178	0.146	-1.22	8	0.259	

Note. *****Conditional on lifealc7 and lifesmk7.

```
***The statement edf= specifies the complete-
***data degrees of freedom for the parameter
***estimates. This is used to compute an
***adjusted degrees of freedom for each
***parameter estimate;
proc mianalyze parms=b.mixparms edf=8;
  modeleffects intercept prog1 prog2 prog3;
  run;
```

All analyses were adjusted for substance use in grade 7 (lifealc7 and lifesmk7). For comparison purposes, similar analyses were conducted with three other MI datasets (dummy coding the cluster variable, ignoring the cluster structure, and a hybrid of these two methods), and also using incomplete data with PROC MIXED (pairwise deletion). In the first MI data set, we assumed fixed cluster effects by including a dummy-coded cluster variable in normal-model MI. In the second MI data set, the cluster structure was ignored from normal-model MI. For the third MI data set, we adopted a hybrid dummy code strategy proposed by Graham (2012), which performs half of the normal-model imputations with fixed effects MI and the other half ignoring the cluster structure. All MI models were conducted with 40 imputations. The analysis made use of all 40 imputed datasets and the parameter estimates were combined across datasets using the "PROC MIANALYZE" procedure" in SAS. The edf=8 (8 is complete data degrees of freedom) option was used to adjust degrees of freedom for inference (Barnard & Rubin, 1999). The ICC (conditional on lifealc7 and lifesmk7) was calculated from the "Covariance Estimates Table" of the SAS output. Parameter estimates and the ICC generated by each of the models were compared across the four different kinds of imputed datasets and the incomplete data (with pairwise deletion).

Results

Table 2 shows parameter estimates for program membership predicting lifetime smoking behavior in grade 9 (lifesmk9) by imputation method. Regression coefficients were generally comparable across imputation methods. When compared with PAN, estimates for the standard error (SE) of program membership predicting "lifesmk9" were higher using dummy-coded MI, and lower using MI ignoring the cluster structure, the hybrid strategy, and pairwise deletion. Correspondingly, p-values and conditional ICC were overestimated with dummy-coded MI and underestimated with MI ignoring the cluster structure, the hybrid strategy, and pairwise deletion.

Different imputation methods still reached the same statistical conclusion that prog1 and prog3 were not significant predictors of "lifesmk9" (Table 2). Prog2 was not a significant predictor of "lifesmk9" using PAN

($p>0.05$) and dummy-coded MI but was a significant predictor when the other methods were used (MI ignoring the cluster structure, hybrid, and pairwise deletion) ($p<0.05$).

Discussion

The missing data issue is common in CRTs and should be dealt with appropriately in order to obtain unbiased parameter estimates. Although SAS has good built-in features for multilevel analysis, its features for multilevel MI are limited, as Proc MI does not allow random cluster effects in the imputation. In this article, we demonstrated a procedure for a researcher to conduct MI with the PAN package in R through SAS, even if the researcher does not have any prior knowledge of R. Using this procedure it is possible for one to conduct empirical analyses in a way that yields less biased parameter estimates, and reduces Type I or Type II errors.

The empirical example demonstrated that parameter estimates were generally comparable across imputation methods. But when compared with PAN, estimates for the SE, p-value, and ICC were biased. Differing statistical conclusions were also drawn in some occasions (prog2 predicting "lifesmk9").

Including dummy-coded cluster variables in the MI model overestimated the SE, p-value, and ICC. Ignoring the cluster structure underestimated the SE, p-value, and ICC. This is consistent with findings by Andridge (2011) and Taljaard et al. (2008). In this example, the variance estimate that was obtained from the dummy-coded MI model seemed to have a small bias when compared with PAN. On the other hand, MI ignoring the cluster structure had a much larger variance bias and resulted in differing statistical conclusions for prog2 predicting "lifesmk9". This was expected with true ICC around 0.01 to 0.02. When the ICC was larger than 0.01, ignoring cluster structure was found to be associated with evident bias (Taljaard et al., 2008) and dummy-coded MI was associated with small bias (Andridge, 2011).

The hybrid strategy was proposed as a method to reduce variance bias with normal model MI, but it did not prove to be an ideal solution. In particular, using this method resulted in drawing a different statistical conclusion for prog2 predicting "lifesmk9" compared with PAN. As mentioned earlier, the true ICC >.01 in this analysis, the dummy-coded MI had a small bias. Thus results with the hybrid strategy were similar to MI ignoring the cluster structure, although the variance bias was slightly reduced compared with MI ignoring the cluster structure.

Pairwise deletion was not a good solution for handling missing data. Using pairwise deletion also altered the statistical conclusion for prog2 predicting "lifesmk9". Unlike dummy-coded MI or MI ignoring the cluster structure, variance estimates were biased in both directions (underestimated and overestimated).

The results of the empirical example were consistent with previous studies demonstrating that missing data in multilevel studies is handled most appropriately by including a random cluster effect in the MI model. MI ignoring the cluster structure, the hybrid strategy, and pairwise deletion all showed evident bias in the variance estimates. In this particular example, the true ICC was greater than .01, thus dummy-coded MI did not seem to introduce a large bias (Andridge, 2011). Dummy-coded MI could cause substantial bias however, especially with a large percentage of missingness, a small cluster size, a small ICC (<.01) and/or a low correlation between the variables to be imputed and the variables without missing data included in the MI model (Andridge, 2011). To get unbiased parameter estimates and statistical conclusions, MI should be conducted by taking the cluster structure into account appropriately. By following the procedure introduced in this article, MI with the PAN package can be easily conducted in SAS.

Authors' notes and acknowledgments

A version of this work was presented at the annual meeting of the Society for Prevention Research, San Francisco, May 2013. This work was supported in part by a pre-doctoral fellowship in the Penn State PAMT (Prevention And Methodology Training) program, NIDA Grant #T32DA017629; Greenberg, PI; Collins and Smith, Co-PIs.

References

Andridge, R.R. (2011). Quantifying the impact of fixed effects modeling of clusters in multiple imputation for cluster randomized trials. *Biometrical Journal, 53*(1), 57-74.

Barnard, J. & Rubin, D.B. (1999). Small-sample degrees of freedom with multiple imputation. *Biometrika* 86, 948–95.

Graham, J. W. (2012). *Missing data: analysis and design*. New York: Springer.

Hansen, W. B., & Graham, J. W. (1991). Preventing

alcohol, marijuana, and cigarette use among adolescents: Peer pressure resistance training versus establishing conservative norms. *Preventive Medicine*, *20*, 414-430.

Murray, D. M. (1998). *Design and analysis of group-randomized trials*. New York: Oxford University Press.

Schafer, J. L. (2001). Multiple imputation with PAN. In L. M. Collins & A. G. Sayer (Eds.), *New methods for the analysis of change* (pp. 357-377). Washington, DC: American Psychological Association.

Schafer, J. L., & Yucel, R. M. (2002). Computational strategies for multivariate linear mixed-effects models with missing values. *Journal of Computational and Graphical Statistics*, *11*, 437-457.

Taljaard, M., Donner, A., and Klar, N. (2008). Imputation of strategies for missing continuous outcomes in cluster randomized trials. *Biometrical Journal 50*, 329-345.

Xin, W. (2012). %PROC_R: A SAS macrothat enables native R programming in the base SAS environment. *Journal of Statistical Software, code snippet, 46*(2), 1-13.

Appendix

The appendix provides three listings. The line numbers given in the margin are for references in the text only and are not part of the listings.

Listing 1 ■ Syntax for preliminary model specification and diagnostic plots generation.

```
1  options varlenchk=nowarn;
2  *****Imputation with PAN package: preliminary model specification and diagnostic plots;
3  %include "C:\Users\xxx\Downloads\Proc_R.sas"; *<- Specify location of "Proc_R.sas";
4  %Proc_R (SAS2R =b.alc, R2SAS =);    *<-Specify dataset to be imputed by replacing "b.alc";
5  cards4;
6  #####Lines below starting with "#" are comments
7  ########################
8  ######Imputation with PAN
9  ########################
10 ######set working directory;
11 setwd("C:/temp")
12 ###rename data
13 data=b.alc
14 ######cluster variable
15 clustername="school"
16
17 ######variables without missing data
18 covname=c("female","cola1","cola2")#<-Add more variables without missing data here as needed
19              #<-Leave blank if there is no variable without missing data (e.g., covname=c())
20
21 ######number of variables entered above with a random slope
22 n2=0 #<-No change is required for random intercepts and fixed slopes model;
23      #<-Replace 0 by no of variables in line 22 with random slope for random slopes model;
24
25 ######variables with missing data
26 yname=c("frndalc","frndsmk", "frndmar","coffee1","coffee2","lifesmk7","rcntsmk7",
27 "lifealc7","rcntalc7", "peeralc7","peersmk7","lifesmk8","rcntsmk8","lifealc8",
28 "rcntalc8", "peeralc8","peersmk8","lifesmk9","rcntsmk9","lifealc9","rcntalc9")      #<-add
29 more variables with missing data here as needed
30
31 v=1  #<- a fixed term for the intercept is included (1) or not included (2)
```

```
burnin=5000  #<- Number of iterations in the burn-in period before imputation
source("C:/Users/xxx/diagnosticplots.r") #<-Specify location of file "diagnosticplots.r";
basepath1="C:/Users/xxx/Desktop/plotsburnin" #<- Specify loc. of burn-in diagnostic plots;

######Past this point, nothing needs to be edited
library(pan)
data=data[order(data[clustername]),]
cluster=data[clustername][,1]
covm=as.matrix(subset(data,select=covname))
y=data[yname]
y=sapply(y,function(x) x=ifelse(x== '.' ,NA,x))
y=as.matrix(y)
seedno=sample(1000000000,1)
n=nrow(data)
int=rep(1,n)
pred=cbind(int,covm)
k=length(yname)
n1=length(covname)
xcol=v:(n1+1)
zcol=1:(n2+1)
prior=list(a=k,Binv=diag(k,k),c=k*(n2+1),Dinv=diag(k*(n2+1),k*(n2+1)))

######burn in
result=pan(y,cluster,pred,xcol,zcol,prior,seed=seedno,iter=burnin)

#################################
##########diagnostic plots for burn in
#################################
diagnosticplots(basepath1,burnin)
q()
;;;;
%Quit;
```

Listing 2 ■ Syntax for determining sufficient number of iterations between imputations.

```
#####################################
######Copy lines 1 to 55 from Listing 1 to here
######Modify line 48 as shown below
v=2  #<-changes to this line may not be necessary; see discussion in the text
######Number of iterations between imputations
da=1000  #<- Specify number of iterations between imputations
basepath2="C:/Users/xxx/Desktop/plotsda" #<- Specify location of diagnostic plots;
result=pan(y,cluster,pred,xcol,zcol,prior,seed=seedno,iter=da,start=result$last)

#######################################
######diagnostic plots for a single imputation
#######################################
diagnosticplots(basepath2,da)
q()
```

```
;;;;
%Quit;
```

Listing 3 ■ Syntax for multiple imputation.

```
########################################
######Copy lines 1 to 57 from Listing 2 to here
######Modify line 4 as shown below
%Proc_R (SAS2R=b.alc, R2SAS=b.pan); #<-Specify imputed dataset by replacing "b.pan";
########################################

########################
######Multiple imputations
########################

######Number of imputations
m=40  #<-Specify number of imputations here

######Including variables not included in imputation (e.g., treatment variable) for further
analysis
othername=c("prog1","prog2","prog3")
other=as.matrix(subset(data,select=othername))

new=NULL
for (i in 1:m){
imputation=i
seedno=sample(1000000000,1)
result=pan(y,cluster,pred,xcol,zcol,prior,seed=seedno,iter=da,start=result$last)
new=rbind(new,cbind(imputation,result$y,cluster,covm,other))
}

new=data.frame(new,row.names=NULL)
names(new)=c("imputation",yname,clustername,covname,othername)
b.pan=new #<-Specify imputed dataset by replacing "b.pan" as in line 4;
;;;;
%Quit;
```

Citation

Zhou, J., Connell, L. E., & Graham, J. W. (2014). Handling missing data in cluster randomized trials: A demonstration of multiple imputation with PAN through SAS *The Quantitative Methods for Psychology*, *10* (2), 153-166.

Copyright © 2014 Zhou, Connell, & Graham. This is an open-access article distributed under the terms of the *Creative Commons Attribution License (CC BY)*. The use, distribution or reproduction in other forums is permitted, provided the original author(s) or licensor are credited and that the original publication in this journal is cited, in accordance with accepted academic practice. No use, distribution or reproduction is permitted which does not comply with these terms.

Received: 24/01/14 ~ Accepted: 1/04/14