Tutorial For Simulated Cluster Randomised Trials

Courtney McDermott

Multilevel Data Structures

Data collected in a clinical trial, or another biomedical study design, can sometimes be "multilevel". This means that the data may be "clustered" in different groups, such as students located within a school. This is a tutorial on how to create a three-level data structure with missingness at any level of the data. We will pretend that this code simulates a longitudinal, cluster-randomised clinical trial of students located within schools. The trial is on a new education tool, and the treatment groups (tool versus control, or no tool) are randomly allocated to the clusters, or schools, rather than the individuals. The outcome measure, a measure of intelligence, is a normally-distributed, continuous variable. This variable has been standardised to have a mean of 0 and standard deviation of 1. Level 3 will represent the repeated outcome measures, which are clustered within the Level 2 units, or students. The students are clustered within the Level 1 units, or schools. For this tutorial, you will need to install and load the following packages: simstudy, nlme, car, dplyr, and faux.

Simulating the data: Level 1 (Clusters)

To start, we will begin by specifying the number of individuals within each cluster. This can be simulated as a fixed number for all clusters by specifying the distribution as **dist = "uniform"**. If you would like the number to vary, you can specify the distribution as **"noZeroPoisson"**. There are many other distributions to choose from. The example below will represent a trial with varying number of individuals within each cluster. Here, we have simulated the data to have approximately 20 students within each school, and 20 schools are participating in the trial.

```
#specified number of students (20) with a Poisson distribution
gen.school <- defData(varname = "nStudents", formula = 20,
                      dist = "noZeroPoisson", id="School")
```

If you would like to have individuals within each school to be slightly correlated with one another, you can increase the intraclass correlation coefficient, or the ICC. One way to do this would be to give all students within each school a small mean shift that is unique to each school. Here this is specified as a normal distribution with a mean of 0 and a standard deviation of 1. We can see that the output is a datatable that describes the two School-level, or Level 1, variables created (nStudents and schoolBaselineShift).

```
#Adding a baseline shift for each school
gen.school <- defData(gen.school, varname = "schoolBaselineShift",
                      dist = "normal", formula = 0, variance = 1)
gen.school
```

```
##                varname formula variance          dist     link
## 1:           nStudents      20        0 noZeroPoisson identity
## 2: schoolBaselineShift       0        1        normal identity
```

Next, we will specify the number of clusters. We will link it back to the above datatable by calling the function **genData()**. This will then produce a dataset of the two above variables, plus an id variable for school. We can now see the number of students in each school and the mean baseline shift that will be applied to the students within each school.

```
#Generate the Level 3 dataset. Specify number of clusters and link
#to the descriptive table from above (gen.school)
dtSchool <- genData(20, gen.school)
dtSchool
```

```
##     School nStudents schoolBaselineShift
##  1:      1        24         -0.99612975
##  2:      2        23         -1.11394990
##  3:      3        19         -0.05573154
##  4:      4        18          1.17443240
##  5:      5        18          1.05321861
##  6:      6        16          0.05760597
##  7:      7        20         -0.73504289
##  8:      8        14          0.93052842
##  9:      9        31          1.66821097
## 10:     10        16          0.55968789
## 11:     11        24         -0.75397477
## 12:     12        21          1.25655419
## 13:     13        26          0.03849255
## 14:     14        25          0.18953983
## 15:     15        32          0.46259495
## 16:     16        26         -0.42736305
## 17:     17        25          0.01658600
## 18:     18        16          0.70487910
## 19:     19        18          0.97184932
## 20:     20        23         -0.62049160
```

Assigning cluster-level covariates

We will now randomly assign each school to a treatment group. It is possible to assign more than 2 groups here. Additionally, one can non-randomly assign the groups by simply using the line:

```
dtSchool$Treatment = c()
```

The user could assign each school to a chosen treatment group by ordering the treatment assignment within the concatenate command.

To assign it randomly, we use the following line of code:

```
#Adding treatment at the level of the cluster, 2 groups, randomly
#assigned to half the schools
dtSchool <- trtAssign(dtSchool, n=2, grpName="Treatment")
```

We can add another cluster-level covariate here, such as the gender of the school. We will call a value of 1 is “all girls”, 2 is “all boys” and 3 is “mixed”. This will be added in a non-random fashion.

```
#Creating School Gender variable at level of cluster
dtSchool$SchoolGender = c(1,2,3,1,2,3,1,2,3,1)
```

Simulating the data: Level 2 (students)

Finally, we will generate the full dataset by telling the programme to add the number of students based on the "nStudents" variable as the identiy variable. The new id variable for students will be labelled "idChild".

```
#generated full dataset, with number of students
dtStudent = genCluster(dtSchool, "School", numIndsVar = "nStudents",
                       level1ID = "idChild")
head(dtStudent)
```

```
##    School Treatment nStudents schoolBaselineShift SchoolGender idChild
## 1:      1         1        24          -0.9961298            1       1
## 2:      1         1        24          -0.9961298            1       2
## 3:      1         1        24          -0.9961298            1       3
## 4:      1         1        24          -0.9961298            1       4
## 5:      1         1        24          -0.9961298            1       5
## 6:      1         1        24          -0.9961298            1       6
```

Simulating the data: Level 3 (repeated measures)

We now have a two-level data structure. To add the third level, the repeated measurements, we will start with creating baseline measurements for each individual. We will create a correlation matrix for the repeated measurements. For this example, we will assume that measurements that are one time-point apart will have a correlation of 0.8, and measurements that are two time-points apart will have a correlation of 0.7. The baseline mean, as mentioned above, is 0 and the standard deviation will be 1.

```
#Correlation matrix for the repeated measures
#Identity matrix on diagonals, since each measurement has a correlation of 1 with itself.
#One time-point apart = 0.8 (1 and 2; 2 and 3); Two time-points apart = 0.7 (1 and 3)
C = matrix(c(1, 0.8, 0.7,
             0.8, 1, 0.8,
             0.7, 0.8, 1), nrow=3)

#created dataset of 3 repeated measures, with no treatment or time effect,
#with length based on student information
baselinedt = genCorData(length(dtStudent$idChild), mu=c(0,0,0),
                        sigma=c(1, 1, 1), corMatrix=C)
```

We now have a dataset of three repeated measurments, with no treatment or time effect. We will create small treatment by time interaction effects to add to those variables, to create our full repeated measures for the main dtStudent dataset. The first measurement will be baseline, so all individuals will have no treatment effect. The second two measurements will represent the first and second follow-up data collection. These will have a Treatment by Time slope of 1. Two random, normally-distributed variables were created with variance of 0.25 to be added to only those in the treatment group.

Simulating the data: full dataset

Next, the repeated measurements, as well as the small treatment effect variables, were combined with the student data. The school mean shifts were added to baseline, and the treatment effects were added to the follow-up variables for those in the treatment group.

```
#create a random variable for Treatment*Time effect for the second
#time-point (first follow-up), Tr1
#for second follow-up, created effect of 2 (still slope of 1), Tr2
gen.student <- defDataAdd(varname = "Tr1", dist = "normal", formula = 1,
                          variance = .25)
gen.student <- defDataAdd(gen.student, varname = "Tr2", dist = "normal",
                          formula = 1, variance = .25)

#combined student/school dataset with above Treatment*Time effects
dtStudent <- addColumns(gen.student, dtStudent)

#Adding the repeated measures to the student dataset
#First is baseline, second two are follow-up 1 ("FU1") and follow-up 2 ("FU2")
dtStudent <- cbind(dtStudent, Baseline = baselinedt$V1)
dtStudent <- cbind(dtStudent, FU1 = baselinedt$V2)
dtStudent <- cbind(dtStudent, FU2 = baselinedt$V3)

#Added the school mean shift to baseline
#Added the Treatment*Time effect for those in the treatment group
#Baseline (plus school baseline shift)
dtStudent$Baseline <- dtStudent$Baseline + dtStudent$schoolBaselineShift
#FU1 (first follow up)
dtStudent$FU1 <- dtStudent$FU1 + dtStudent$schoolBaselineShift +
  (dtStudent$Tr1*dtStudent$Treatment)
#FU2 (second follow up)
dtStudent$FU2 <- dtStudent$FU2 + dtStudent$schoolBaselineShift +
  (dtStudent$Tr1*dtStudent$Treatment) + (dtStudent$Tr2*dtStudent$Treatment)


#Reducing variables to just those that are needed
dtStudent1 <- dtStudent[, c("School", "idChild","Treatment", "SchoolGender",
                            "Baseline", "FU1", "FU2")]
```

Analysing the full analysis set: linear mixed modelling

We now want to get the Treatment by Time interaction effect, and we will do this by running a linear mixed model. A linear mixed model can be specified many ways, with fixed or random effects. Here, we will specify a linear mixed model with a random slope for each school, and each student within the school. The "weights" command will allow us to give a random variance for each school. Three fixed-effect predictors will be included in the model: Treatment, Time, and Treatment by Time interaction. Finally, we will specify the correlation structure of the repeated measures over Time to be "Compound Symmetry", meaning only one variance and covariance will be estimated for all pairs of measurements. **The lme()** function allows users to specify other types of correlation structures, such as autoregressive or unstructured.

The choice to specify whether an effect is random or fixed, at the cluster or individual level, is up to the user, and is dependent upon your specific data structure. Similarly, the correlation structure will need to be chosen by the user and is dependent upon the data.

To include Time as a predictor in nlme, we first we will have to change the data to long format. Next, we will change the new "Time" variable into "1", "2", and "3" for each time-point. It will be changed to an integer so that we can get one slope for the Treatment*Time effect over the three time-points. Finally, we will run the linear mixed model and investigate the results. This dataset, before any missing data is introduced, will be referred to as the "full analysis set" or FAS.

```
#Changing to long format
dtStudent_long <- melt(dtStudent1,
                       # ID variables - all the variables to keep but not split apart on
                       id.vars=c("School", "Treatment", "idChild", "SchoolGender"),
                       # The source columns
                       measure.vars=c("Baseline", "FU1", "FU2" ),
                       # Name of the destination column that will identify the original
                       # column that the measurement came from
                       variable.name="Time",
                       value.name="Outcome")

#recoding into numbers
dtStudent_long$Time = recode(dtStudent_long$Time, " 'Baseline'=1")
dtStudent_long$Time = recode(dtStudent_long$Time, " 'FU1'=2")
dtStudent_long$Time = recode(dtStudent_long$Time, " 'FU2'=3")

#specifying Time as an integer rather than a factor
dtStudent_long$Time = as.integer(dtStudent_long$Time)


#running linear mixed model to determine coefficients of full analysis set (fas)
#specified a random intercept by school
#specified compound symmetry covariance structure
lmm_fas <- lme(Outcome ~ Time + Treatment + Treatment*Time,
            random = list(School = ~1, idChild = ~1),
            weights =varIdent(School),
            corr = corCompSymm(form= ~Time),
            data=dtStudent_long, na.action="na.omit",
            method = "REML")
```

Results of the linear mixed model

We will now look at the results of this linear mixed model. The full results can be found using the **summary()** function. First, we will look at the number of observations in the FAS, which is equivalent to the total number of measurements (number of students x number of measures, or 3). We will then look at the number of students and the number of schools. Finally, we will look at estimated treatment, time, and treatment by time effects. The code below demonstrates how to get these values.

```
#looking at total number of "N", or observations
#(number of students x number of measurements, or 3)
summary(lmm_fas)$dims$N

#Looking at number of children and number of schools
summary(lmm_fas)$dims$ngrps

#Looking at coefficients
summary(lmm_fas)
```

The total number of observations is 1305. This is from 435 students (with three measurements) in 20 schools. The Treatment*Time effect is 0.984 (SE = 0.039), which is close to the simulated effect of 1.

Introducing Missingness: Level 3

So we now have determined the Treatment over Time effect of this simulated intervention. The next step is to introduce missingness into the dataset, re-analyse, and compare the results to the FAS results.

We will first introduce missingness at Level 3, or the repeated measurements. For this tutorial, we will simulate 20% missingness overall. To introduce missingness in the repeated measures, the data will need to be in wide format. The first missing data mechanism will be missing completely at random, or completely unrelated to anything. First, an object will be created called "Del" that will be a vector of 0s and 1s based on our instructions. For missing completely at random, this will be created using the **sample()** function. The probability of a 0, or being missing, is 20% and the probability of a 1, or being observed, is 80%.

After this vector is created, we will use it to introduce missingness. When Del = 0, the observation will be replaced with an "NA", or a blank space. When Del = 1, the observation will remain in the dataset as is. The Del vector is created three times, once per outcome measurment, to simulate non-monotone missingness. This means that the probability of missingness at one time-point is not correlated with the probability of missingness at another time-point. An example of this may be if people have conflicts on one of the data collection days and therefore miss getting their data collected. However, they are present for the other data collection days. Another example may be if the instrument used to collect particpant information randomly glitched for 20% of the individuals on any given data collection day.

```
#create object that contains the length of the dataset in wide format
nlength = length(dtStudent1$Baseline)
#create blank "Del" object, length of dataset
Del = rep(NA, nlength)

#probability of missing is 20% or 0.20. Would be 0.10 for 10% or 0.40 for 40%.
#this is missing completely at random
Del= sample(c(0,1), size = nlength, prob=c(0.2, 0.8), replace=T) #with replacement
#Creating new variable with missingness where Del=0
dtStudent1$Baseline_20MCAR = ifelse(Del==0, NA, dtStudent1$Baseline)

#performing same for FU1
Del= sample(c(0,1), size = nlength, prob=c(0.2, 0.8), replace=T)
#Creating new variable with missingness where Del=0
dtStudent1$FU1_20MCAR = ifelse(Del==0, NA, dtStudent1$FU1)

#performing same for FU2
Del= sample(c(0,1), size = nlength, prob=c(0.2, 0.8), replace=T)
#Creating new variable with missingness where Del=0
dtStudent1$FU2_20MCAR = ifelse(Del==0, NA, dtStudent1$FU2)

#creating new dataset of just new deleted variables
Data1 = dtStudent1[, c("School", "idChild", "Treatment", "Baseline_20MCAR",
                       "FU1_20MCAR", "FU2_20MCAR", "SchoolGender")]
```

We now have a new dataset with the missingness introduced. Approximately 20% of the observations are missing completely at random. We will transform this into long format and re-run our linear mixed model to see how this missingness impacts the estimated Treatment by Time effect. We will get the summary statistics using the **summary()** functions. The code below presents how to get this information.

```r
#changing to long format
Data1_long <- melt(Data1,
                   # ID variables - all the variables to keep but not split apart on
                   id.vars=c("School", "idChild", "Treatment", "SchoolGender"),
                   # The source columns
                   measure.vars=c("Baseline_20MCAR", "FU1_20MCAR", "FU2_20MCAR" ),
                   # Name of the destination column that will identify the original
                   # column that the measurement came from
                   variable.name="Time",
                   value.name="Outcome")

Data1_long$Time = recode(Data1_long$Time, " 'Baseline_20MCAR'=1")
Data1_long$Time = recode(Data1_long$Time, " 'FU1_20MCAR'=2")
Data1_long$Time = recode(Data1_long$Time, " 'FU2_20MCAR'=3")

#changing Time into an integer rather than factor
Data1_long$Time = as.integer(Data1_long$Time)

#performing linear mixed on data with missingness, same model as Full analysis set
lmm_level3mcar <- lme(Outcome ~ Time + Treatment + Treatment*Time,
              random = list(School = ~1, idChild = ~1),
              weights =varIdent(School),
              corr = corCompSymm(form= ~Time),
              data=Data1_long, na.action="na.omit",
              method = "REML",
              control=(msMaxIter=100000))

#looking at total number of "N", or observations
#(number of students x number of measurements, or 3)
summary(lmm_level3mcar)$dims$N

#Looking at number of children and number of schools
summary(lmm_level3mcar)$dims$ngrps

#Looking at coefficients
summary(lmm_level3mcar)
```

The results of this new analysis show that the total number of observations is 1038, compared to 1305 from the full analysis set. A linear mixed model employs maximum likelihood, so it loses information from the lost measures, as seen in this loss of observations. However, this method only drops *individuals* from the analysis if they do not have any recorded observations. Here, after missingness is introduced, the number of children is 430, compared to 435 from the FAS. That means that only 5 students were missing all three observations. The number of schools remains at 20 since at least one child was fully observed in each school.

We can see that the Treatment*Time effect has changed slightly from 0.984 to 0.978 after the deletions. The standard error has increased from 0.039 to 0.045. This is due to the loss of information from the missing observations.

Introducing Missingness: Level 2

As stated above, Level 2 observations are the individuals, or students. If data are missing at Level 2, it means that the data for the entire student are missing for all three measurements.

To do this, we will again use the wide format of the FAS data. We will repeat the process of sampling 0,1 for the number of students in the dataset. However, any 0 will mean that all of the data will be deleted for that individual.

```
#create object that contains the length of the dataset in wide format
nlength = length(dtStudent1$Baseline)
#create blank "Del" object, length of dataset
Del = rep(NA, nlength)


#probability of missing is 20% or 0.20. Would be 0.10 for 10% or 0.40 for 40%.
#this is missing completely at random
Del= sample(c(0,1), size = nlength, prob=c(0.2, 0.8), replace=T) #with replacement

#Creating new variables with missingness where Del=0
#Will be missing for all three measurments
dtStudent1$Baseline_20MCAR2 = ifelse(Del==0, NA, dtStudent1$Baseline)
dtStudent1$FU1_20MCAR2 = ifelse(Del==0, NA, dtStudent1$FU1)
dtStudent1$FU2_20MCAR2 = ifelse(Del==0, NA, dtStudent1$FU2)


#creating new dataset of just new deleted variables
Data2 = dtStudent1[, c("School", "idChild", "Treatment", "Baseline_20MCAR2",
                       "FU1_20MCAR2", "FU2_20MCAR2", "SchoolGender")]
```

In this new dataset, Data2, 20% of the students are completely missing all data. As with the Level 3 missingness, we will re-run the linear mixed model analysis and compare to the results of the full analysis set. The code below demonstrates how to get the summary statistics.

```
#changing to long format
Data2_long <- melt(Data2,
                   # ID variables - all the variables to keep but not split apart on
                   id.vars=c("School", "idChild", "Treatment", "SchoolGender"),
                   # The source columns
                   measure.vars=c("Baseline_20MCAR2", "FU1_20MCAR2", "FU2_20MCAR2"),
                   # Name of the destination column that will identify the original
                   # column that the measurement came from
                   variable.name="Time",
                   value.name="Outcome")

Data2_long$Time = recode(Data2_long$Time, " 'Baseline_20MCAR2'=1")
Data2_long$Time = recode(Data2_long$Time, " 'FU1_20MCAR2'=2")
Data2_long$Time = recode(Data2_long$Time, " 'FU2_20MCAR2'=3")

#changing Time into an integer rather than factor
Data2_long$Time = as.integer(Data2_long$Time)

#performing linear mixed on data with missingness, same model as FAS
lmm_level2mcar <- lme(Outcome ~ Time + Treatment + Treatment*Time,
             random = list(School = ~1, idChild = ~1),
             weights =varIdent(School),
             corr = corCompSymm(form= ~Time),
             data=Data2_long, na.action="na.omit",
             method = "REML",
```

```
             control=(msMaxIter=100000))


#looking at total number of "N", or observations
#(number of students x number of measurements, or 3)
summary(lmm_level2mcar)$dims$N

#Looking at number of children and number of schools
summary(lmm_level2mcar)$dims$ngrps

#Looking at coefficients
summary(lmm_level2mcar)
```

After the Level 2 data are deleted, there are now only 343 students in the analysis, compared to 435 in the FAS and 430 from the Level 3 deletions. However,the analysis includes 1029 observations, which is comparable to the Level 3 deletions. The analysis also still includes 20 schools. The Treatment by Time effect has increased to 1.00 from 0.984 from the FAS. The standard error has increased to 0.043, due to the drop in sample size, but this is slightly smaller than the missing level 3 data analysis.

Introducing Missingness: Level 1

As stated above, Level 1 observations are the clusters, or schools. If data are missing at Level 1, it means that the data for the entire cluster are missing for all three measurements.

To do this, we will again use the wide format of the FAS data. We will repeat the process of sampling 0,1, except this time it will be for the number of *clusters* in the dataset. The easiest way to do this would be to create a new variable called Del, using the **sample()** function, at the very beginning of the simulations when the data only include Level 1 data. However, we will perform this using the full dataset in wide format using a loop function.

```
#create object that contains the length of the number of schools/clusters
nlength = length(unique(dtStudent1$School))
#create blank "Del" object, length of dataset
Del = rep(NA, nlength)

#probability of missing is 20% or 0.20. Would be 0.10 for 10% or 0.40 for 40%.
#this is missing completely at random
Del= sample(c(0,1), size = nlength, prob=c(0.2, 0.8), replace=T)

#ordering by School number
dtStudent1 = dtStudent1[order(dtStudent1$School),]

#creating blank variable
dtStudent1$Del = rep(NA, length(dtStudent1$idChild))

#creating loop to fill in the Del value for entire school
for(j in unique(dtStudent1$School)){
  dtStudent1$Del[dtStudent1$School==j] = Del[j]  }
```

So at this point, we have a new variable in the dataset called “Del” that is the same value for the entire school. If a school has a Del value of 1, the data remain in the dataset. If the school has a Del value of 0, the data will be deleted for that entire school.

```
dtStudent1$Baseline_20MCAR3 = ifelse(dtStudent1$Del==0, NA, dtStudent1$Baseline)
dtStudent1$FU1_20MCAR3 = ifelse(dtStudent1$Del==0, NA, dtStudent1$FU1)
dtStudent1$FU2_20MCAR3 = ifelse(dtStudent1$Del==0, NA, dtStudent1$FU2)


#creating new dataset of just new deleted variables
Data3 = dtStudent1[, c("School", "idChild", "Treatment", "Baseline_20MCAR3",
                       "FU1_20MCAR3", "FU2_20MCAR3", "SchoolGender")]
```

In this new dataset, Data2, 20% of the students are completely missing all data. As with the Level 3 missingness, we will re-run the linear mixed model analysis and compare to the results of the full analysis set. The code below demonstrates how to run this analysis and determine the summary statistics.

```
#changing to long format
Data3_long <- melt(Data3,
                   # ID variables - all the variables to keep but not split apart on
                   id.vars=c("School", "idChild", "Treatment", "SchoolGender"),
                   # The source columns
                   measure.vars=c("Baseline_20MCAR3", "FU1_20MCAR3", "FU2_20MCAR3" ),
                   # Name of the destination column that will identify the original
                   # column that the measurement came from
                   variable.name="Time",
                   value.name="Outcome")

Data3_long$Time = recode(Data3_long$Time, " 'Baseline_20MCAR3'=1")
Data3_long$Time = recode(Data3_long$Time, " 'FU1_20MCAR3'=2")
Data3_long$Time = recode(Data3_long$Time, " 'FU2_20MCAR3'=3")

#changing Time into an integer rather than factor
Data3_long$Time = as.integer(Data3_long$Time)

#performing linear mixed on data with missingness, same model as Full analysis set
lmm_level1mcar <- lme(Outcome ~ Time + Treatment + Treatment*Time,
             random = list(School = ~1, idChild = ~1),
             weights =varIdent(School),
             corr = corCompSymm(form= ~Time),
             data=Data3_long, na.action="na.omit",
             method = "REML",
             control=(msMaxIter=100000))


#looking at total number of "N", or observations
#(number of students x number of measurements, or 3)
summary(lmm_level1mcar)$dims$N

#Looking at number of children and number of schools
summary(lmm_level1mcar)$dims$ngrps

#Looking at coefficients
summary(lmm_level1mcar)
```

After the Level 1 deletions, the number of observations is 1041. This is comparable to the analyses with missing data in Level 2 and Level 3. The total number of children is 347, which is again comparable to the

previous two analyses. However, the difference here is that only 16 schools are included in the analysis. The Treatment by Time effect has been reduced to 0.943 and the standard error is inflated to 0.043.

Overall, we see that there is not a *huge* difference in the results for MCAR missingness at any level. It appears that that the Level 1 missingness resulted in the largest Treatment by Time effect bias, but this could be due to simulation error and over 1000 simulates, the three analyses are fairly comparable. The biases are very small for the treatment effect and the standard errors are slightly increased due to the loss of information in the datasets.

Other missing data mechanisms

Two other patterns exist in which the missing data can occur. The following sections will provide the code to generate each type of missingness in each level but the analyses will not be re-run.

Missing at Random

Data will be missing at random if they are missing dependent upon an observed covariate. The covariate in this tutorial will be the "SchoolGender" variable. For missingness in all three levels, we will simulate it so that a SchoolGender value of 1 corresponds to a missingness probability of 0.1; SchoolGender value of 2 corresponds to a missingness probability of 0.2; and SchoolGender value of 3 corresponds to a probability of 0.3. This will end up with an approximate overall missingness rate of 0.2, or 20%.

Level 3 Missingness Code:

```
#First, order dataset by SchoolGender
dtStudent1 = dtStudent1[order(dtStudent1$SchoolGender),]

#create blank "Del" object, length of dataset
Del = rep(NA, nlength)

#create three separate missingness probabilities that correspond to the
#SchoolGender variable: 10%, 20%, and 30%
Del= c(sample(c(0,1), length(dtStudent1$SchoolGender[dtStudent1$SchoolGender==1]),
              prob=c(0.1, 0.9), replace=T),
       sample(c(0,1), length(dtStudent1$SchoolGender[dtStudent1$SchoolGender==2]),
              prob=c(0.2, 0.8), replace=T),
       sample(c(0,1), length(dtStudent1$SchoolGender[dtStudent1$SchoolGender==3]),
              prob=c(0.3, 0.7), replace=T))

#Creating new variable with missingness where Del=0
dtStudent1$Baseline_20MAR = ifelse(Del==0, NA, dtStudent1$Baseline)


#performing same for FU1
Del= c(sample(c(0,1), length(dtStudent1$SchoolGender[dtStudent1$SchoolGender==1]),
              prob=c(0.1, 0.9), replace=T),
       sample(c(0,1), length(dtStudent1$SchoolGender[dtStudent1$SchoolGender==2]),
              prob=c(0.2, 0.8), replace=T),
       sample(c(0,1), length(dtStudent1$SchoolGender[dtStudent1$SchoolGender==3]),
              prob=c(0.3, 0.7), replace=T))
```

```
#Creating new variable with missingness where Del=0
dtStudent1$FU1_20MAR = ifelse(Del==0, NA, dtStudent1$FU1)


#performing same for FU2
Del= c(sample(c(0,1), length(dtStudent1$SchoolGender[dtStudent1$SchoolGender==1]),
              prob=c(0.1, 0.9), replace=T),
       sample(c(0,1), length(dtStudent1$SchoolGender[dtStudent1$SchoolGender==2]),
              prob=c(0.2, 0.8), replace=T),
       sample(c(0,1), length(dtStudent1$SchoolGender[dtStudent1$SchoolGender==3]),
              prob=c(0.3, 0.7), replace=T))
#Creating new variable with missingness where Del=0
dtStudent1$FU2_20MAR = ifelse(Del==0, NA, dtStudent1$FU2)


#creating new dataset of just new deleted variables
Data1 = dtStudent1[, c("School", "idChild", "Treatment", "Baseline_20MAR",
                       "FU1_20MAR", "FU2_20MAR", "SchoolGender")]
```

Level 2 Missingness Code:

```
#First, order dataset by SchoolGender
dtStudent1 = dtStudent1[order(dtStudent1$SchoolGender),]

#create blank "Del" object, length of dataset
Del = rep(NA, nlength)

#create three separate missingness probabilities that correspond to the
#SchoolGender variable: 10%, 20%, and 30%
Del= c(sample(c(0,1), length(dtStudent1$SchoolGender[dtStudent1$SchoolGender==1]),
              prob=c(0.1, 0.9), replace=T),
       sample(c(0,1), length(dtStudent1$SchoolGender[dtStudent1$SchoolGender==2]),
              prob=c(0.2, 0.8), replace=T),
       sample(c(0,1), length(dtStudent1$SchoolGender[dtStudent1$SchoolGender==3]),
              prob=c(0.3, 0.7), replace=T))

#Creating new variable with missingness where Del=0
dtStudent1$Baseline_20MAR2 = ifelse(Del==0, NA, dtStudent1$Baseline)
dtStudent1$FU1_20MAR2 = ifelse(Del==0, NA, dtStudent1$FU1)
dtStudent1$FU2_20MAR2 = ifelse(Del==0, NA, dtStudent1$FU2)


#creating new dataset of just new deleted variables
Data2 = dtStudent1[, c("School", "idChild", "Treatment", "Baseline_20MAR2",
                       "FU1_20MAR2", "FU2_20MAR2", "SchoolGender")]
```

Level 1 Missingness Code:

For level 1, we will actually go back to the original Level 1 dataset, **dtSchool**. We will then perform a similar function to create this "Del" object where schools are missing dependent upon SchoolGender value. Then, using a loop function, this will be applied to the dtStudent1 dataset.

```
#create object that contains the length of the number of schools/clusters
nlength = length(dtSchool$School)
#create blank "Del" object, length of dataset
Del = rep(NA, nlength)

#ordering by SchoolGender
dtSchool = dtSchool[order(dtSchool$SchoolGender),]


Del= c(sample(c(0,1), length(dtSchool$SchoolGender[dtSchool$SchoolGender==1]),
              prob=c(0.1, 0.9), replace=T),
       sample(c(0,1), length(dtSchool$SchoolGender[dtSchool$SchoolGender==2]),
              prob=c(0.2, 0.8), replace=T),
       sample(c(0,1), length(dtSchool$SchoolGender[dtSchool$SchoolGender==3]),
              prob=c(0.3, 0.7), replace=T))

#ordering dtStudent1 dataset by SchoolGender
dtStudent1 = dtStudent1[order(dtStudent1$SchoolGender),]

#creating loop to fill in the Del value for entire school
for(j in unique(dtStudent1$School)){
  dtStudent1$Del[dtStudent1$School==j] = Del[j]  }

dtStudent1$Baseline_20MAR3 = ifelse(dtStudent1$Del==0, NA, dtStudent1$Baseline)
dtStudent1$FU1_20MAR3 = ifelse(dtStudent1$Del==0, NA, dtStudent1$FU1)
dtStudent1$FU2_20MAR3 = ifelse(dtStudent1$Del==0, NA, dtStudent1$FU2)

#creating new dataset of just new deleted variables
Data3 = dtStudent1[, c("School", "idChild", "Treatment", "Baseline_20MAR3",
                       "FU1_20MAR3", "FU2_20MAR3", "SchoolGender")]
```

Missing Not at Random

The next few subsections will focus on data that are missing not at random. This means that the probability of missingness is directly correlated with the values themselves. Rather than creating just one or a few probabilities to sample from, each outcome value/individual/cluster will have a probability of missingess. This means that the Del vector will be created by simulating a variable of probabilities, ranging from 0 to 1 with a mean of 0.2 (for 20% missingness), that has a positive correlation coefficient of 0.5 with the outcome values. As a result, the variables/individuals/clusters with larger outcome values will be more likely to be missing.

Level 3 Missingness Code:

```
#first, sorting dtStudent1 by Baseline values
dtStudent1 = dtStudent1[order(dtStudent1$Baseline),]

#creating probability vector with mean of 0.20, standard deviation of 0.05,
#and correlation coefficient of 0.5. Correlated with Baseline
MNARprobMiss = rnorm_pre(dtStudent1$Baseline, mu=.20, sd=.05, r=0.5)
MNARprobObs = 1-MNARprobMiss
```

```
nlength = length(dtStudent1$Baseline)
Del=rep(NA, nlength)

#creating loop to sample 0,1 based on the above probability vector for
#each baseline value
for (j in 1:nlength)
{Del[j] = sample(c(0,1), size = 1, prob=c(MNARprobMiss[j], MNARprobObs[j]))
}

dtStudent1$Baseline_20MNAR = ifelse(Del==0, NA, dtStudent1$Baseline)


#sorting by FU1 values this time
dtStudent1 = dtStudent1[order(dtStudent1$FU1),]
#Correlated with FU1
MNARprobMiss = rnorm_pre(dtStudent1$FU1, mu=.20, sd=.05, r=0.5)
MNARprobObs = 1-MNARprobMiss
Del=rep(NA, nlength)

#creating loop to sample 0,1 based on the above probability vector for
#each FU1 value
for (j in 1:nlength)
{Del[j] = sample(c(0,1), size = 1, prob=c(MNARprobMiss[j], MNARprobObs[j]))  }

dtStudent1$FU1_20MNAR = ifelse(Del==0, NA, dtStudent1$FU1)


#sorting by FU2 values this time
dtStudent1 = dtStudent1[order(dtStudent1$FU2),]
#Correlated with FU2
MNARprobMiss = rnorm_pre(dtStudent1$FU2, mu=.20, sd=.05, r=0.5)
MNARprobObs = 1-MNARprobMiss
Del=rep(NA, nlength)


for (j in 1:nlength)
{Del[j] = sample(c(0,1), size = 1, prob=c(MNARprobMiss[j], MNARprobObs[j]))  }

dtStudent1$FU2_20MNAR = ifelse(Del==0, NA, dtStudent1$FU2)

#creating new dataset of just new deleted variables
Data3 = dtStudent1[, c("School", "idChild", "Treatment", "Baseline_20MNAR",
                       "FU1_20MNAR", "FU2_20MNAR", "SchoolGender")]
```

Now, this will inevitably result in more individuals missing in the intervention group for FU1 and FU2 variables, as they are more likely to have larger values with the treatment effect. These vectors can be manipulated how the user sees fit, or split the data into two, based on treatment group, and specify probability vectors for each group.

Level 2 Missingness Code

For the level 2 MNAR mechanism, we will just use the FU2 values. Larger FU2 values will be correlated with a larger probability that the entire individual will be missing.

```
#first, sorting dtStudent1 by Baseline values
dtStudent1 = dtStudent1[order(dtStudent1$FU2),]

#creating probability vector with mean of 0.20, standard deviation of 0.05,
#and correlation coefficient of 0.5. Correlated with FU2
MNARprobMiss = rnorm_pre(dtStudent1$FU2, mu=.20, sd=.05, r=0.5)
MNARprobObs = 1-MNARprobMiss

nlength = length(dtStudent1$FU2)
Del=rep(NA, nlength)

#creating loop to sample 0,1 based on the above probability vector for
#each FU2 value
for (j in 1:nlength)
{Del[j] = sample(c(0,1), size = 1, prob=c(MNARprobMiss[j], MNARprobObs[j]))
}

#Deleting based on Del values
dtStudent1$Baseline_20MNAR2 = ifelse(Del==0, NA, dtStudent1$Baseline)
dtStudent1$FU1_20MNAR2 = ifelse(Del==0, NA, dtStudent1$FU1)
dtStudent1$FU2_20MNAR2 = ifelse(Del==0, NA, dtStudent1$FU2)

#creating new dataset of just new deleted variables
Data3 = dtStudent1[, c("School", "idChild", "Treatment", "Baseline_20MNAR2",
                       "FU1_20MNAR2", "FU2_20MNAR2", "SchoolGender")]
```

Level 1 Missingness Code

As above, this can be simulated many ways. For this example, we will simulate the probability of missingness to be correlated with the average cluster Treatment by Time effect. First, we will calculate each individual's difference from Baseline to FU2. We will determine the average difference per cluster, and then simulate the probability vector to be correlated with these cluster averages.

Another method to simulate Missing Not at Random would be to just hand select the schools/data with higher outcome values to be missing, rather than creating a correlated probability vector. The method presented here allows for a bit of "randomness". However, since these are simulations, they do allow the user to simulate the missingness mechanism however they sit fit. For instance, you could hand select the four schools with the highest FU2 average to be missing–as a "most extreme" missingness mechanism–and see how extreme the results change compared to the FAS results.

```
#order by school
dtStudent1 = dtStudent1[order(dtStudent1$School)]

#create new variable of differences
dtStudent1$trtDiff = dtStudent1$FU2 - dtStudent1$Baseline

#create blank vector
ClusterMean = rep(NA, length(unique(dtStudent1$School)))

#creating loop to fill in the ClusterMean value for each school
for(j in 1:length(unique(dtStudent1$School))){
  ClusterMean[j] = mean(dtStudent1$FU2[dtStudent1$School==j])  }
```

```r
#ordering it by school and creating new variable
dtSchool = dtSchool[order(dtSchool$School)]
dtSchool$ClusterMean=ClusterMean

library(faux)
library(dplyr)

#creating correlated probability vector with the ClusterMean variable
MNARprobMiss = rnorm_pre(dtSchool$ClusterMean, mu=.20, sd=.05, r=0.5)
MNARprobObs = 1-MNARprobMiss

dtSchool$MNARprobMiss = MNARprobMiss
dtSchool$MNARprobObs = MNARprobObs

Del = rep(NA, length(dtSchool$School))

#creating loop to sample 0,1 based on the above probability vector for
#each ClusterMean value
for (j in 1:length(dtSchool$School))
{Del[j] = sample(c(0,1), size = 1, prob=c(dtSchool$MNARprobMiss[j],
                                          dtSchool$MNARprobObs[j]))  }

dtSchool$Del = Del
#creating loop to fill in the Del value for entire school
for(j in unique(dtStudent1$School)){
  dtStudent1$Del[dtStudent1$School==j] = Del[j]  }


#Deleting based on Del values
dtStudent1$Baseline_20MNAR3 = ifelse(dtStudent1$Del==0, NA, dtStudent1$Baseline)
dtStudent1$FU1_20MNAR3 = ifelse(dtStudent1$Del==0, NA, dtStudent1$FU1)
dtStudent1$FU2_20MNAR3 = ifelse(dtStudent1$Del==0, NA, dtStudent1$FU2)

#creating new dataset of just new deleted variables
Data3 = dtStudent1[, c("School", "idChild", "Treatment", "Baseline_20MNAR3",
                       "FU1_20MNAR3", "FU2_20MNAR3", "SchoolGender")]
```

Final Guidance

This tool can be used to customise the number of repeated measures, number of individuals, number of clusters, number of individuals within clusters, level of missingness (or missingness in multiple levels!), percent missing, and missing data mechanism. The code can be manipulated to fit any 3-level data with continuous outcomes. The code can also be manipulated so that the outcome variables are binary or categorical. The purpose of this tutorial is to help guide individuals in simulating 3-level data with missingness in any or all levels to determine how this will impact the results of the trial.