qbs121_hw6_gibran

Gibran Erlangga

2/21/2022

GEE

Using the data sets you used for for Week 5 for LMM and GLMM models:

1. Refit the models for both LMM and GLMM using generalized estimating equations:

```r
library(tidyverse)
library(geepack)
library(lme4)
library(mice)

# LMM
data <- read.csv('HousePrices.csv')

# data preprocessing
boolean_convert <- function(data) {
  if (data == "yes") {
    return(1)
  } else {
    return(0)
  }
}

data$prefer <- sapply(data$prefer, boolean_convert)

df <- data %>%
      select('price', 'lotsize', 'bedrooms', 'stories', 'garage', 'prefer')

# binary
data_binary <- read.csv('https://stats.idre.ucla.edu/stat/data/hdp.csv')
```

1a. Use the family/link function assumptions and working correlation structures that closest to the assumptions used in the previous LMM and GLMM fits. For instance, a random cluster effect in LMM/GLMM would be closest to an exchangeable working correlation matrix. Contrast the fitted coefficients for fixed effects from the GEE models to those you got from LMM and GLMM, and comment on any differences that might be seen between the coefficient estimates and standard errors.

```r
# modeling
model <- lmer(log(price) ~ log(lotsize) + bedrooms + garage + stories +
                (1 | prefer), data = df)
print(summary(model))
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: log(price) ~ log(lotsize) + bedrooms + garage + stories + (1 |
##     prefer)
##    Data: df
##
## REML criterion at convergence: 66.8
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.3665 -0.6408  0.0572  0.6145  2.7973
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  prefer   (Intercept) 0.01627  0.1275
##  Residual             0.06244  0.2499
## Number of obs: 546, groups:  prefer, 2
##
## Fixed effects:
##              Estimate Std. Error t value
## (Intercept)   7.27369    0.26446  27.504
## log(lotsize)  0.39387    0.02962  13.299
## bedrooms      0.07422    0.01610   4.609
## garage        0.07185    0.01339   5.366
## stories       0.12618    0.01353   9.326
##
## Correlation of Fixed Effects:
##             (Intr) lg(lt) bedrms garage
## log(lotsiz) -0.924
## bedrooms    -0.079 -0.066
## garage       0.301 -0.340 -0.098
## stories      0.035 -0.059 -0.399  0.035
```

```
model_gee <- geeglm(log(price) ~ log(lotsize) + bedrooms + garage + stories,
                    id=prefer, family=gaussian, data = df)
print(summary(model_gee))
```

```
##
## Call:
## geeglm(formula = log(price) ~ log(lotsize) + bedrooms + garage +
##     stories, family = gaussian, data = df, id = prefer)
##
##  Coefficients:
##              Estimate  Std.err   Wald Pr(>|W|)
## (Intercept)  6.875502 0.223410 947.12  < 2e-16 ***
## log(lotsize) 0.433564 0.026173 274.41  < 2e-16 ***
## bedrooms     0.078892 0.019673  16.08 6.07e-05 ***
## garage       0.072795 0.020323  12.83 0.000341 ***
## stories      0.126298 0.008707 210.42  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation structure = independence
## Estimated Scale Parameters:
##
```

```
##              Estimate Std.err
## (Intercept)  0.06742 0.00539
## Number of clusters:   7  Maximum cluster size: 336
```

```
model_binary <- glmer(remission ~ IL6 + CRP + CancerStage + Experience +
    (1 | DID), data = data_binary, family = binomial, nAGQ = 10)
print(summary(model_binary))
```

```
## Generalized linear mixed model fit by maximum likelihood (Adaptive
##   Gauss-Hermite Quadrature, nAGQ = 10) [glmerMod]
##  Family: binomial  ( logit )
## Formula: remission ~ IL6 + CRP + CancerStage + Experience + (1 | DID)
##    Data: data_binary
##
##      AIC      BIC   logLik deviance df.resid
##     7408     7465    -3696     7392     8517
##
## Scaled residuals:
##    Min     1Q Median     3Q    Max
## -3.531 -0.444 -0.199  0.401  7.142
##
## Random effects:
##  Groups Name        Variance Std.Dev.
##  DID    (Intercept) 4.04     2.01
## Number of obs: 8525, groups:  DID, 407
##
## Fixed effects:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -2.6264     0.5066   -5.18  2.2e-07 ***
## IL6              -0.0564     0.0115   -4.90  9.6e-07 ***
## CRP              -0.0223     0.0102   -2.18    0.029 *
## CancerStageII    -0.4874     0.0730   -6.68  2.4e-11 ***
## CancerStageIII   -1.1259     0.0924  -12.19  < 2e-16 ***
## CancerStageIV    -2.5088     0.1508  -16.64  < 2e-16 ***
## Experience        0.1193     0.0274    4.35  1.3e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) IL6    CRP    CncSII CnSIII CncSIV
## IL6         -0.082
## CRP         -0.097  0.000
## CancerStgII -0.069  0.009  0.000
## CancrStgIII -0.047  0.013  0.010  0.446
## CancerStgIV -0.020  0.036  0.009  0.282  0.249
## Experience  -0.963 -0.006 -0.002 -0.008 -0.014 -0.019
```

```
model_binary_gee <- geeglm(remission ~ IL6 + CRP + CancerStage + Experience,
                           id=DID, data = data_binary, family = binomial)
print(summary(model_binary_gee))
```

```
##
## Call:
```

```
## geeglm(formula = remission ~ IL6 + CRP + CancerStage + Experience,
##     family = binomial, data = data_binary, id = DID)
##
##  Coefficients:
##                 Estimate  Std.err   Wald Pr(>|W|)
## (Intercept)     -1.73707  0.35147  24.43  7.7e-07 ***
## IL6             -0.03823  0.00807  22.44  2.2e-06 ***
## CRP             -0.01774  0.00785   5.11    0.024 *
## CancerStageII   -0.33205  0.05863  32.08  1.5e-08 ***
## CancerStageIII  -0.71077  0.07920  80.53  < 2e-16 ***
## CancerStageIV   -1.72709  0.12925 178.57  < 2e-16 ***
## Experience       0.08401  0.01923  19.09  1.2e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation structure = independence
## Estimated Scale Parameters:
##
##             Estimate Std.err
## (Intercept)    0.996  0.0574
## Number of clusters:   407  Maximum cluster size: 40
```

Between the lmer() and geeglm() model for continuous outcome above, the coefficients differ by varying degree. For the intercept, bedrooms, garage and stories variables, the coefficient we got from lmer() is higher than geeglm(), while the log(lotsize) coefficient is lower in lmer() compared to geeglm(). For standard error values between the two models, all variables have lower standard errors except for bedrooms and garage variables.

For the binary outcome, the coefficients of intercept, IL6, CRP, CancerStageIII and CancerStageIV are higher in geeglm() compared to glmer(). For the standard errors, all variables have lower standard error values in geeglm() compared to glmer() except for CancerStageII.

1.b Use the alternative working correlation structures available in GEE to see if that results in are any further differences in the estimated regression coefficients and standard errors.

```r
model_gee_exc <- geeglm(log(price) ~ log(lotsize) + bedrooms + garage + stories,
                        id=prefer, family=gaussian, data = df,
                        corstr='exchangeable')
print(summary(model_gee_exc))
```

```
##
## Call:
## geeglm(formula = log(price) ~ log(lotsize) + bedrooms + garage +
##     stories, family = gaussian, data = df, id = prefer, corstr = "exchangeable")
##
##  Coefficients:
##               Estimate Std.err   Wald Pr(>|W|)
## (Intercept)    7.54609 0.21003 1290.8  < 2e-16 ***
## log(lotsize)   0.36814 0.02358  243.8  < 2e-16 ***
## bedrooms       0.07376 0.01526   23.4  1.4e-06 ***
## garage         0.07346 0.02065   12.7  0.00037 ***
## stories        0.12161 0.00876  192.6  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Correlation structure = exchangeable
## Estimated Scale Parameters:
##
##             Estimate Std.err
## (Intercept)   0.0769  0.0125
##   Link = identity
##
## Estimated Correlation Parameters:
##       Estimate Std.err
## alpha    0.259   0.155
## Number of clusters:   7  Maximum cluster size: 336
```

```
model_gee_ar1 <- geeglm(log(price) ~ log(lotsize) + bedrooms + garage + stories,
                        id=prefer, family=gaussian, data = df,
                        corstr='ar1')
print(summary(model_gee_ar1))
```

```
##
## Call:
## geeglm(formula = log(price) ~ log(lotsize) + bedrooms + garage +
##     stories, family = gaussian, data = df, id = prefer, corstr = "ar1")
##
##  Coefficients:
##              Estimate Std.err    Wald Pr(>|W|)
## (Intercept)   9.20123 0.25564 1295.53  < 2e-16 ***
## log(lotsize)  0.18708 0.02086   80.42  < 2e-16 ***
## bedrooms      0.05402 0.00705   58.76  1.8e-14 ***
## garage        0.04236 0.02200    3.71    0.054 .
## stories       0.06165 0.01286   22.97  1.6e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation structure = ar1
## Estimated Scale Parameters:
##
##             Estimate Std.err
## (Intercept)   0.0866 0.00416
##   Link = identity
##
## Estimated Correlation Parameters:
##       Estimate Std.err
## alpha    0.854  0.0541
## Number of clusters:   7  Maximum cluster size: 336
```

```
model_binary_gee_exc <- geeglm(remission ~ IL6 + CRP + CancerStage + Experience,
                               id=DID, data = data_binary, family = binomial,
                               corstr='exchangeable')
print(summary(model_binary_gee_exc))
```

```
##
## Call:
## geeglm(formula = remission ~ IL6 + CRP + CancerStage + Experience,
##     family = binomial, data = data_binary, id = DID, corstr = "exchangeable")
##
##  Coefficients:
##                 Estimate  Std.err   Wald Pr(>|W|)
## (Intercept)    -1.43713  0.33023  18.94  1.3e-05 ***
## IL6            -0.03588  0.00685  27.42  1.6e-07 ***
## CRP            -0.01420  0.00639   4.94  0.02617 *
## CancerStageII  -0.30598  0.04746  41.57  1.1e-10 ***
## CancerStageIII -0.70791  0.06750 109.98  < 2e-16 ***
## CancerStageIV  -1.63002  0.11811 190.45  < 2e-16 ***
## Experience      0.06502  0.01778  13.37  0.00026 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation structure = exchangeable
## Estimated Scale Parameters:
##
##              Estimate Std.err
## (Intercept)    0.981   0.047
##   Link = identity
##
## Estimated Correlation Parameters:
##       Estimate Std.err
## alpha    0.368  0.0368
## Number of clusters:   407  Maximum cluster size: 40
```

```r
model_binary_gee_ar1 <- geeglm(remission ~ IL6 + CRP + CancerStage + Experience,
                               id=DID, data = data_binary, family = binomial,
                               corstr='ar1')
print(summary(model_binary_gee_ar1))
```

```
##
## Call:
## geeglm(formula = remission ~ IL6 + CRP + CancerStage + Experience,
##     family = binomial, data = data_binary, id = DID, corstr = "ar1")
##
##  Coefficients:
##                 Estimate  Std.err   Wald Pr(>|W|)
## (Intercept)    -1.06814  0.52148   4.20    0.041 *
## IL6            -0.05095  0.00867  34.56  4.1e-09 ***
## CRP            -0.01040  0.00691   2.27    0.132
## CancerStageII  -0.29748  0.05529  28.95  7.4e-08 ***
## CancerStageIII -0.72336  0.08229  77.28  < 2e-16 ***
## CancerStageIV  -1.46257  0.12250 142.55  < 2e-16 ***
## Experience      0.04654  0.02824   2.72    0.099 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation structure = ar1
## Estimated Scale Parameters:
##
##              Estimate Std.err
## (Intercept)    0.972   0.053
##   Link = identity
```

```
##
## Estimated Correlation Parameters:
##       Estimate Std.err
## alpha    0.876  0.0143
## Number of clusters:   407  Maximum cluster size: 40
```

I tried two correlation structures from GEE: exchangeable and independence.

Seeing the results of geeglm() with exchangeable as the working correlation structure for continuous outcome, I observed similar pattern as the default geeglm() result from previous question except for the intercept coefficient, where it has a higher coefficient value in geeglm() compared to lmer(). For standard error values between the two models, all variables have lower standard errors in geeglm() except for garage variable.

Seeing the results of geeglm() with ar1 as the working correlation structure for continuous outcome, the intercept and stories variables have higher coefficient values in geeglm() compared to lmer(), while the rest of the variables have lower coefficient values in geeglm() compared to lmer(). For standard error values between the two models, all variables have lower standard errors in geeglm() except for garage variable.

Looking at the results of geeglm() with exchangeable as the working correlation structure for binary outcome, all variables have higher coefficient values in geeglm() compared to glmer() except for Experience variable. For standard error values between the two models, all variables have lower standard errors in geeglm() compared to glmer().

Looking at the results of geeglm() with ar1 as the working correlation structure for binary outcome, all variables have higher coefficient values in geeglm() compared to glmer(). For standard error values between the two models, all variables have lower standard errors in geeglm() compared to glmer(), except for the intercept and Experience variables.

MISSING DATA: MICE

2. For missing data, choose online data sets suitable for both binary and continous outcome regressions involving multiple predictor variables using glm.

Data Dictionary:
GPA - First-year college GPA on a 0.0 to 4.0 scale
HSGPA - High school GPA on a 0.0 to 4.0 scale
SATV - Verbal/critical reading SAT score
SATM - Math SAT score
Male - 1= male, 0= female
HU - Number of credit hours earned in humanities courses in high school
SS - Number of credit hours earned in social science courses in high school
FirstGen - 1= student is the first in her or his family to attend college, 0=otherwise
White - 1= white students, 0= others
CollegeBound - 1=attended a high school where >=50% students intended to go on to college, 0=otherwise

```r
data <- read.csv('FirstYearGPA.csv')
```

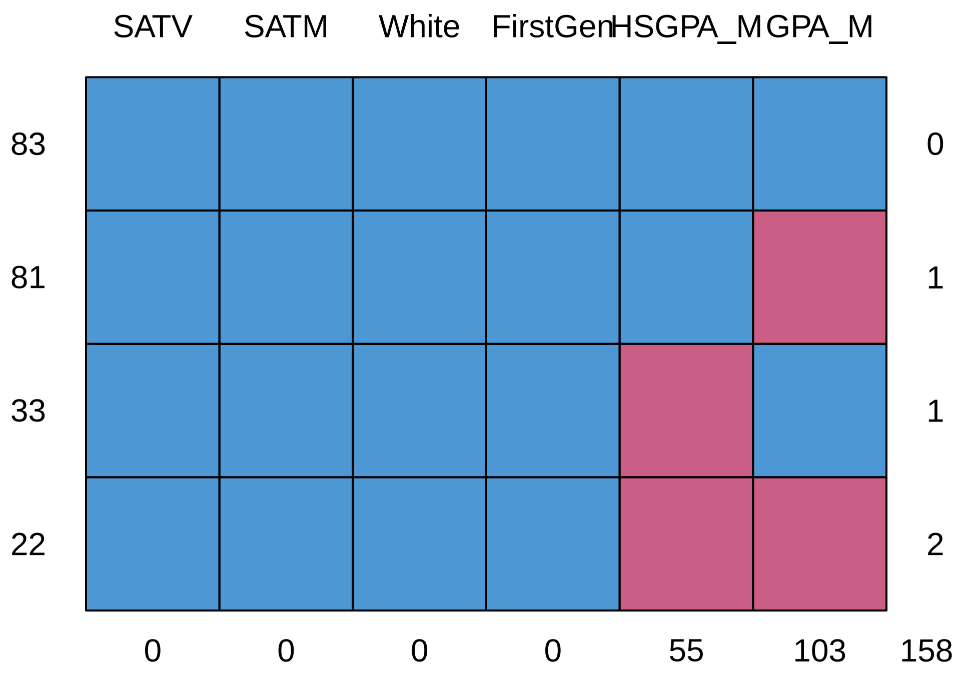
2.b Using similar techniques discussed in class, create MAR and MCAR missingness in the outcome variable. These are specified in the .rmd file posted for the class. For MAR, this involves assigning outcome or predictor variable values to be missing depending on the value of an variable that is completely nonmissing. For MCAR, this can be just randomly assigning values of the outcome variable to be missing.

```r
n <- dim(data)[1]

# missing at random (MAR)
# A case where the probability of being missing is the same only within
# groups defined by the observed data
selectht<-data$HU<median(data$HU)
ns<-sum(selectht)
data$HSGPA_M<-data$HSGPA
data$HSGPA_M[selectht][runif(ns)<0.5] <- NA

# Missing Completely at Random (MCAR)
# A case where the probability of being missing is the same for all cases
data$GPA_M <- ifelse(runif(n)<0.5, NA, data$GPA)

# plot missing data
data_model <- data[, c("GPA_M", "HSGPA_M", "SATV", "SATM", "White", "FirstGen")]
md.pattern(data_model)
```



```
##    SATV SATM White FirstGen HSGPA_M GPA_M
## 83    1    1     1        1       1     1   0
## 81    1    1     1        1       1     0   1
## 33    1    1     1        1       0     1   1
## 22    1    1     1        1       0     0   2
##       0    0     0        0      55   103 158
```

```r
# model on no missing data
print(summary(lm(GPA ~ HSGPA + SATV + SATM + White + FirstGen, data=data)))
```

```
##
## Call:
## lm(formula = GPA ~ HSGPA + SATV + SATM + White + FirstGen, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.0583 -0.2763  0.0307  0.2831  0.8767
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.45e-01   3.21e-01    2.32   0.0214 *
## HSGPA        5.22e-01   7.38e-02    7.07  2.2e-11 ***
## SATV         6.56e-04   4.05e-04    1.62   0.1062
## SATM        -7.60e-06   4.21e-04   -0.02   0.9856
## White        2.21e-01   7.17e-02    3.08   0.0023 **
## FirstGen    -1.57e-01   8.88e-02   -1.77   0.0786 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.395 on 213 degrees of freedom
## Multiple R-squared:  0.295,  Adjusted R-squared:  0.278
## F-statistic: 17.8 on 5 and 213 DF,  p-value: 9.7e-15
```

```r
# model with MAR
print(summary(lm(GPA ~ HSGPA_M + SATV + SATM + White + FirstGen, data=data)))
```

```
##
## Call:
## lm(formula = GPA ~ HSGPA_M + SATV + SATM + White + FirstGen,
##     data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.1207 -0.2478  0.0547  0.2652  0.7019
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  8.62e-01   3.60e-01    2.39    0.018 *
## HSGPA_M      5.29e-01   8.23e-02    6.43  1.5e-09 ***
## SATV        -8.73e-07   4.88e-04    0.00    0.999
## SATM         5.01e-04   4.77e-04    1.05    0.295
## White        2.03e-01   8.21e-02    2.47    0.015 *
## FirstGen    -1.25e-01   1.13e-01   -1.11    0.269
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.388 on 158 degrees of freedom
##   (55 observations deleted due to missingness)
## Multiple R-squared:  0.277,  Adjusted R-squared:  0.255
## F-statistic: 12.1 on 5 and 158 DF,  p-value: 5.88e-10
```

```r
# model with MAR + MCAR
print(summary(lm(GPA_M ~ HSGPA_M + SATV + SATM + White + FirstGen, data=data)))
```

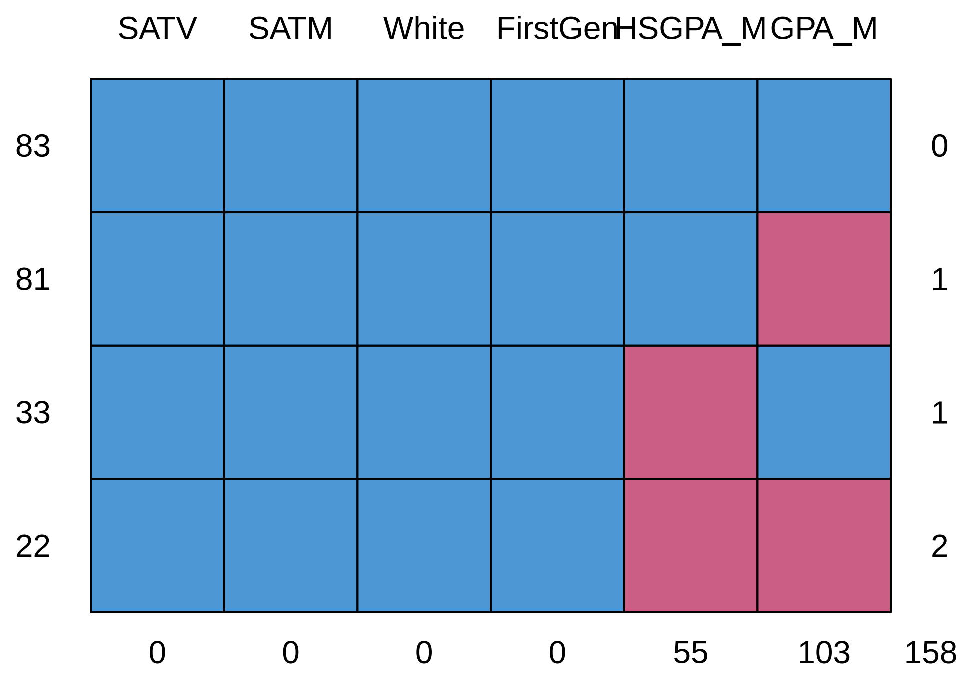
```
##
## Call:
## lm(formula = GPA_M ~ HSGPA_M + SATV + SATM + White + FirstGen,
##     data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.1088 -0.2777  0.0234  0.2504  0.7685
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  8.06e-01   5.46e-01    1.48    0.144
## HSGPA_M      5.70e-01   1.12e-01    5.10  2.4e-06 ***
## SATV        -9.13e-05   7.05e-04   -0.13    0.897
## SATM         2.90e-04   8.01e-04    0.36    0.718
## White        2.89e-01   1.43e-01    2.01    0.048 *
## FirstGen    -2.22e-02   2.01e-01   -0.11    0.913
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.417 on 77 degrees of freedom
##   (136 observations deleted due to missingness)
## Multiple R-squared:  0.304,  Adjusted R-squared:  0.258
## F-statistic: 6.71 on 5 and 77 DF,  p-value: 3.07e-05
```

Fit the models with and without multiple imputation and compare the results. Also suggest how to create (not missing at random) NMAR missingness, and outline how this might be implemented. This need not be programmed.

```r
#data_raw
print(summary(lm(GPA_M ~ HSGPA_M + SATV + SATM + White + FirstGen, data=data)))
```

```
##
## Call:
## lm(formula = GPA_M ~ HSGPA_M + SATV + SATM + White + FirstGen,
##     data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.1088 -0.2777  0.0234  0.2504  0.7685
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  8.06e-01   5.46e-01    1.48    0.144
## HSGPA_M      5.70e-01   1.12e-01    5.10  2.4e-06 ***
## SATV        -9.13e-05   7.05e-04   -0.13    0.897
## SATM         2.90e-04   8.01e-04    0.36    0.718
## White        2.89e-01   1.43e-01    2.01    0.048 *
## FirstGen    -2.22e-02   2.01e-01   -0.11    0.913
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.417 on 77 degrees of freedom
##   (136 observations deleted due to missingness)
## Multiple R-squared:  0.304,  Adjusted R-squared:  0.258
## F-statistic: 6.71 on 5 and 77 DF,  p-value: 3.07e-05
```

```
# plot missing data
md.pattern(data_model)
```



```
##    SATV SATM White FirstGen HSGPA_M GPA_M
## 83    1    1     1        1       1     1   0
## 81    1    1     1        1       1     0   1
## 33    1    1     1        1       0     1   1
## 22    1    1     1        1       0     0   2
##       0    0     0        0      55   103 158
```

```
# MICE method
imput_mice <- mice(data_model, m=5)
```

```
##
##  iter imp variable
##   1   1  GPA_M  HSGPA_M
##   1   2  GPA_M  HSGPA_M
##   1   3  GPA_M  HSGPA_M
##   1   4  GPA_M  HSGPA_M
##   1   5  GPA_M  HSGPA_M
##   2   1  GPA_M  HSGPA_M
##   2   2  GPA_M  HSGPA_M
##   2   3  GPA_M  HSGPA_M
##   2   4  GPA_M  HSGPA_M
##   2   5  GPA_M  HSGPA_M
##   3   1  GPA_M  HSGPA_M
##   3   2  GPA_M  HSGPA_M
##   3   3  GPA_M  HSGPA_M
##   3   4  GPA_M  HSGPA_M
##   3   5  GPA_M  HSGPA_M
##   4   1  GPA_M  HSGPA_M
##   4   2  GPA_M  HSGPA_M
##   4   3  GPA_M  HSGPA_M
##   4   4  GPA_M  HSGPA_M
##   4   5  GPA_M  HSGPA_M
##   5   1  GPA_M  HSGPA_M
##   5   2  GPA_M  HSGPA_M
##   5   3  GPA_M  HSGPA_M
##   5   4  GPA_M  HSGPA_M
##   5   5  GPA_M  HSGPA_M
```

```
imput_mice_fit <- with(data=imput_mice, exp=lm(GPA_M ~ HSGPA_M + SATV + SATM +
                                                 White + FirstGen))
summary(pool(imput_mice_fit))
```

```
##          term  estimate std.error statistic    df p.value
## 1 (Intercept)  1.214636  0.522214     2.326  9.83 0.04275
## 2     HSGPA_M  0.551571  0.127962     4.310  8.60 0.00218
## 3        SATV  0.000334  0.000665     0.502  9.61 0.62676
## 4        SATM -0.000664  0.000504    -1.317 46.23 0.19444
## 5       White  0.242913  0.095033     2.556 21.83 0.01807
## 6    FirstGen -0.149140  0.188495    -0.791  6.04 0.45875
```

MNAR means that the probability of being missing varies for reasons that are unknown to us. To create Not Missing at Random (NMAR) missing values, we should do the following steps: 1. Choose the pattern of missingness 2. Choose the variable(s) to be applied the missingness pattern on 3. Choose the base distribution to be applied to the variable(s) with the chosen pattern of missingness. Ideally this should not be a uniform distribution, because we want the probability of missingness on each row to be different with each other (e.g. log-normal distribution)

Covariate Measurement Error

2.c In the same models, add substantial additive measurement error to a continuous predictor variable and see how estimates are affected for the variable of interest and for the other variables in the model. This can be done using the rnorm() statement in R.

```
# add measurement error on continuous variable
 data$HSGPA_error_noise <- data$HSGPA + rnorm(219, 0.1, 1)

model <- lm(GPA ~ HSGPA + SATV + SATM + White + FirstGen, data=data)
```

```r
model_noise <- lm(GPA ~ HSGPA_error_noise + SATV + SATM + White + FirstGen,
                  data=data)
summary(model)
```

```
##
## Call:
## lm(formula = GPA ~ HSGPA + SATV + SATM + White + FirstGen, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.0583 -0.2763  0.0307  0.2831  0.8767
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.45e-01   3.21e-01    2.32   0.0214 *
## HSGPA        5.22e-01   7.38e-02    7.07  2.2e-11 ***
## SATV         6.56e-04   4.05e-04    1.62   0.1062
## SATM        -7.60e-06   4.21e-04   -0.02   0.9856
## White        2.21e-01   7.17e-02    3.08   0.0023 **
## FirstGen    -1.57e-01   8.88e-02   -1.77   0.0786 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.395 on 213 degrees of freedom
## Multiple R-squared:  0.295,  Adjusted R-squared:  0.278
## F-statistic: 17.8 on 5 and 213 DF,  p-value: 9.7e-15
```

```r
summary(model_noise)
```

```
##
## Call:
## lm(formula = GPA ~ HSGPA_error_noise + SATV + SATM + White +
##     FirstGen, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.0000 -0.3117  0.0416  0.3289  1.2204
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)        1.98e+00   2.86e-01    6.93  4.9e-11 ***
## HSGPA_error_noise  6.72e-02   2.74e-02    2.45   0.0149 *
## SATV               1.22e-03   4.37e-04    2.78   0.0059 **
## SATM              -2.39e-05   4.67e-04   -0.05   0.9593
## White              2.09e-01   7.86e-02    2.66   0.0085 **
## FirstGen          -7.19e-02   9.66e-02   -0.74   0.4578
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.433 on 213 degrees of freedom
## Multiple R-squared:  0.153,  Adjusted R-squared:  0.133
## F-statistic: 7.71 on 5 and 213 DF,  p-value: 1.11e-06
```