Stat 571 HW5

Dongyang Wang

2023-02-28

```
setwd("/Users/dongyangwang/Desktop/UW/Stat 571/HW/HW 5")
rm(list=ls())
set.seed(42)
```

Question 1

The data generation process partially imitates the approach provided in https://stats.stackexchange.com/questions/184741/how-to-simulate-the-different-types-of-missing-data.

Generally, I created a dataframe of 100 subjects with 10 observations for each subject, putting 7% of the response variables to missingness based on the three ways of missing values. The response variables are obtained in a similar manner as in HW1, where I generated random effects and the error term and got the reponse variables with given parameters of beta after random generation of x. x follows random normal with mean 250 and sd 10. Error and random effects follow normal with mean 0 and sd 10. I then simulate 1000 times, and for each time, I perform LMM and GEE on each kind of missingness. This procedure is done for both complete and available cases.

```
##### generic data setup:

ni      = 100  # 100 subjects
nj      =  10  # 10 observation for each subject
prop.m = .07  # 7% missingness
beta0 = 0.5
beta1 = 1.5

simu_miss=function(ni,nj){
  x    = rnorm(ni*nj, mean=250, sd=10)
  time    = rep(1:nj, times=ni)
  id      = rep(1:ni, each=nj)

  e = rnorm(ni*nj, mean=0, sd=10)
  b = rep(rnorm(ni, mean = 0, sd = 10),nj)
  y       = beta0 + beta1 * x + b + e

  # MCAR
  mcar    = runif(ni*nj, min=0, max=1)
  y.mcar = ifelse(mcar<prop.m, NA, y)  # unrelated to anything

  # MAR
  y.mar = matrix(y, ncol=nj, nrow=ni, byrow=TRUE)
  for(i in 1:ni){
    for(j in 4:nj){
      dif1 = y.mar[i,j-2]-y.mar[i,j-3]
```

```
      dif2 = y.mar[i,j-1]-y.mar[i,j-2]
      if(dif1>0 & dif2>0){  # if weight goes up twice, drops out
        y.mar[i,j:nj] = NA;  break
      }
    }
  }
  y.mar = as.vector(t(y.mar))

  # NMAR
  sort.y = sort(y, decreasing=TRUE)
  nmar   = sort.y[ceiling(prop.m*length(y))]
  y.nmar = ifelse(y>nmar, NA, y)  # doesn't show up when heavier
  return(data.frame(id, time, x, y, y.mcar,y.mar,y.nmar))
}

library(geepack)
library(lme4)

gen.one=function(ni,nj, case = 1){
  df=simu_miss(ni,nj)
  df1 = df
  df2 = df
  df3 = df

  # complete
  if (case == 1){

    id1 = unique(df[is.na(df$y.mcar), ]$id)
    df1 = df[!(df$id %in% id1), ]

    id2 = unique(df[is.na(df$y.mar), ]$id)
    df2 = df[!(df$id %in% id1), ]

    id3 = unique(df[is.na(df$y.nmar), ]$id)
    df3 = df[!(df$id %in% id1), ]

  }

  # available
  #if (case == 2){
  #  df = na.omit(df)
  #}


  # lmm
  lmm1 = lmer(y.mcar ~ x + (1|id), data = df1, REML = T)
  lmm2 = lmer(y.mar ~ x + (1|id), data = df2, REML = T)
  lmm3 = lmer(y.nmar ~ x + (1|id), data = df3, REML = T)

  # gee
  gee1 = geeglm(y.mcar ~ x, id = id, data = df1, corstr = "exchangeable")
  gee2 = geeglm(y.mar ~ x, id = id, data = df2, corstr = "exchangeable")
  gee3 = geeglm(y.nmar ~ x, id = id, data = df3, corstr = "exchangeable")
```

```
  # Estimate variance for efficiency
  lmm1_var0 = vcov(lmm1)[1,1]
  lmm1_var1 = vcov(lmm1)[2,2]
  lmm2_var0 = vcov(lmm2)[1,1]
  lmm2_var1 = vcov(lmm2)[2,2]
  lmm3_var0 = vcov(lmm3)[1,1]
  lmm3_var1 = vcov(lmm3)[2,2]

  gee1_var0 = vcov(gee1)[1,1]
  gee1_var1 = vcov(gee1)[2,2]
  gee2_var0 = vcov(gee2)[1,1]
  gee2_var1 = vcov(gee2)[2,2]
  gee3_var0 = vcov(gee3)[1,1]
  gee3_var1 = vcov(gee3)[2,2]

  # Estimate bias
  lmm1_bias0 = fixef(lmm1)[1] - beta0
  lmm1_bias1 = fixef(lmm1)[1] - beta1
  lmm2_bias0 = fixef(lmm2)[1] - beta0
  lmm2_bias1 = fixef(lmm2)[1] - beta1
  lmm3_bias0 = fixef(lmm3)[1] - beta0
  lmm3_bias1 = fixef(lmm3)[1] - beta1

  gee1_bias0 = coef(gee1)[1] - beta0
  gee1_bias1 = coef(gee1)[2] - beta1
  gee2_bias0 = coef(gee2)[1] - beta0
  gee2_bias1 = coef(gee2)[2] - beta1
  gee3_bias0 = coef(gee3)[1] - beta0
  gee3_bias1 = coef(gee3)[2] - beta1

  return(data.frame( lmm1_var0, lmm1_var1, lmm2_var0, lmm2_var1,
                     lmm3_var0, lmm3_var1, gee1_var0, gee1_var1,
                     gee2_var0, gee2_var1, gee3_var0, gee3_var1,
                     lmm1_bias0, lmm1_bias1, lmm2_bias0, lmm2_bias1,
                     lmm3_bias0, lmm3_bias1, gee1_bias0, gee1_bias1,
                     gee2_bias0, gee2_bias1, gee3_bias0, gee3_bias1
  ) )
}
```

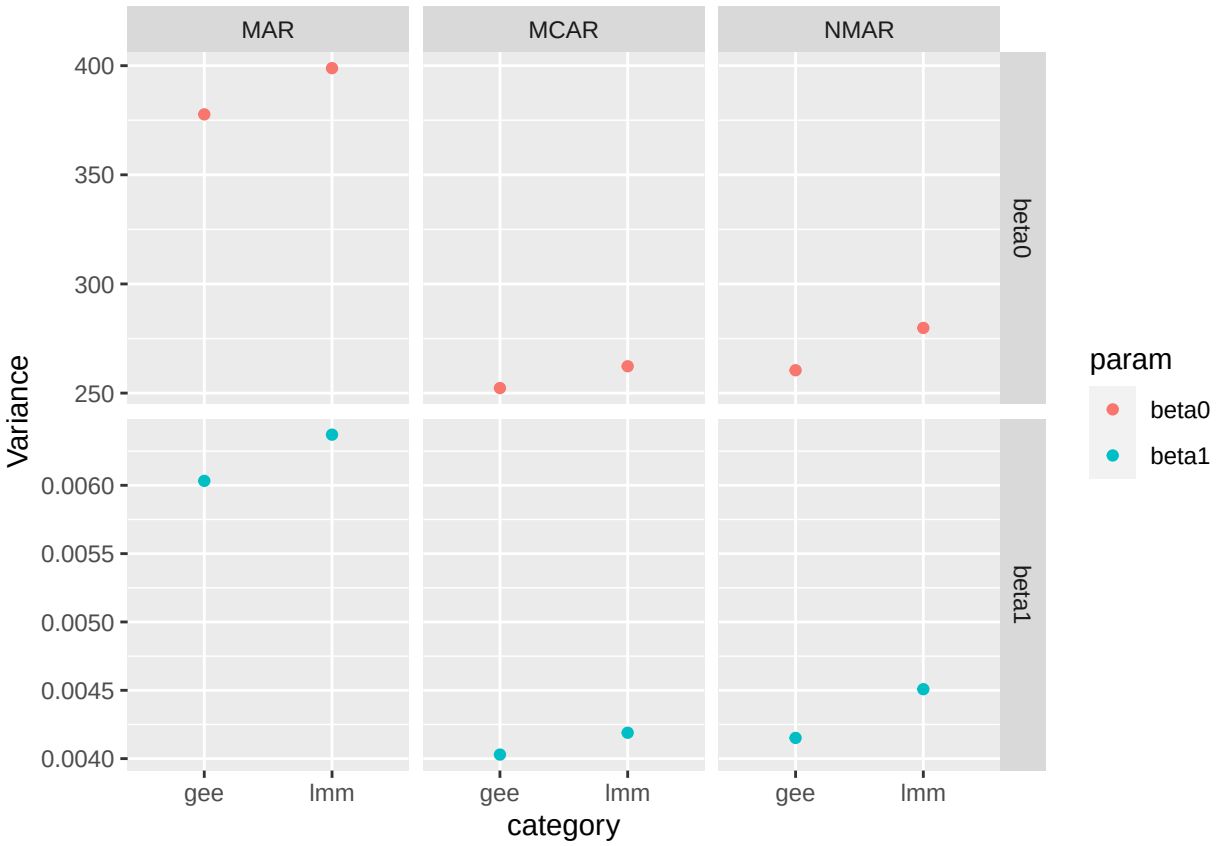
The following code provides a complete case analysis.

```
nrep = 1000

res <- do.call(rbind, lapply(c(1:nrep), function(nrep){
  gen.one(ni,nj, case = 1)
}))
mean_res <- colMeans(res)

simulation_res = as.data.frame(mean_res)
simulation_res
```

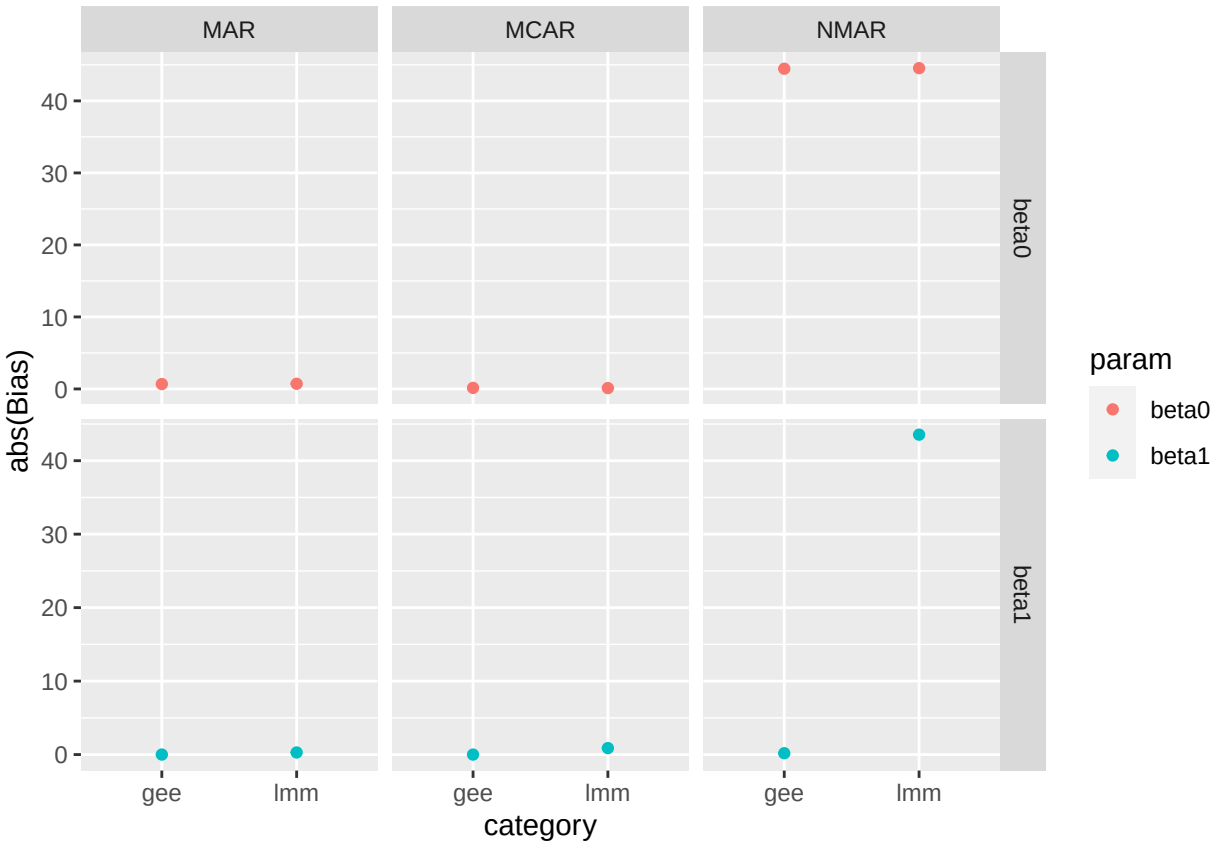
```
##                 mean_res
## lmm1_var0   2.622997e+02
## lmm1_var1   4.189348e-03
```

```
## lmm2_var0   3.988631e+02
## lmm2_var1   6.370670e-03
## lmm3_var0   2.798476e+02
## lmm3_var1   4.508029e-03
## gee1_var0   2.523101e+02
## gee1_var1   4.029603e-03
## gee2_var0   3.776949e+02
## gee2_var1   6.032551e-03
## gee3_var0   2.604800e+02
## gee3_var1   4.151419e-03
## lmm1_bias0  1.203680e-01
## lmm1_bias1 -8.796320e-01
## lmm2_bias0  7.138167e-01
## lmm2_bias1 -2.861833e-01
## lmm3_bias0  4.454567e+01
## lmm3_bias1  4.354567e+01
## gee1_bias0  1.439244e-01
## gee1_bias1 -5.290196e-04
## gee2_bias0  6.752855e-01
## gee2_bias1 -2.712700e-03
## gee3_bias0  4.446704e+01
## gee3_bias1 -1.840543e-01
```

```r
category = c(rep(c("lmm", "gee"), 2, each = 6))
number = c(rep(c("MCAR", "MAR", "NMAR"), 4, each = 2))
metric = c(rep(c("variance", "bias"), each = 12))
param = c(rep(c("beta0", "beta1"),  12))

names(simulation_res)[1] = "value"
simulation_res$category = category
simulation_res$number = number
simulation_res$metric = metric
simulation_res$param = as.factor(param)

simulation_res_var = simulation_res[simulation_res$metric == "variance",]
names(simulation_res_var)[1] = "Variance"
simulation_res_bias = simulation_res[simulation_res$metric != "variance",]
names(simulation_res_bias)[1] = "Bias"

library(ggplot2)
ggplot(data=simulation_res_var, aes(x=category, y=Variance, color = param))+geom_point()+
  facet_grid(cols=vars(number), vars(param), scales="free")
```




```
ggplot(data=simulation_res_bias, aes(x=category, y=abs(Bias), color = param))+geom_point()+
  facet_grid(cols=vars(number), vars(param), scales="free")
```



As we can easily observe from the visualization above, the bias shows a trend.NMAR shows high bias for both models in terms of beta0, but gee performs well in beta1. For other cases, lmm might perform slightly worse but in both the MAR and MCAR cases the bias is negligible.

As for variance, MAR has highest variance for both parameters of all cases, followed by NMAR and then MCAR. LMM seems always to have a higher variance than GEE in terms of beta0 ande beta1 across all missingness.

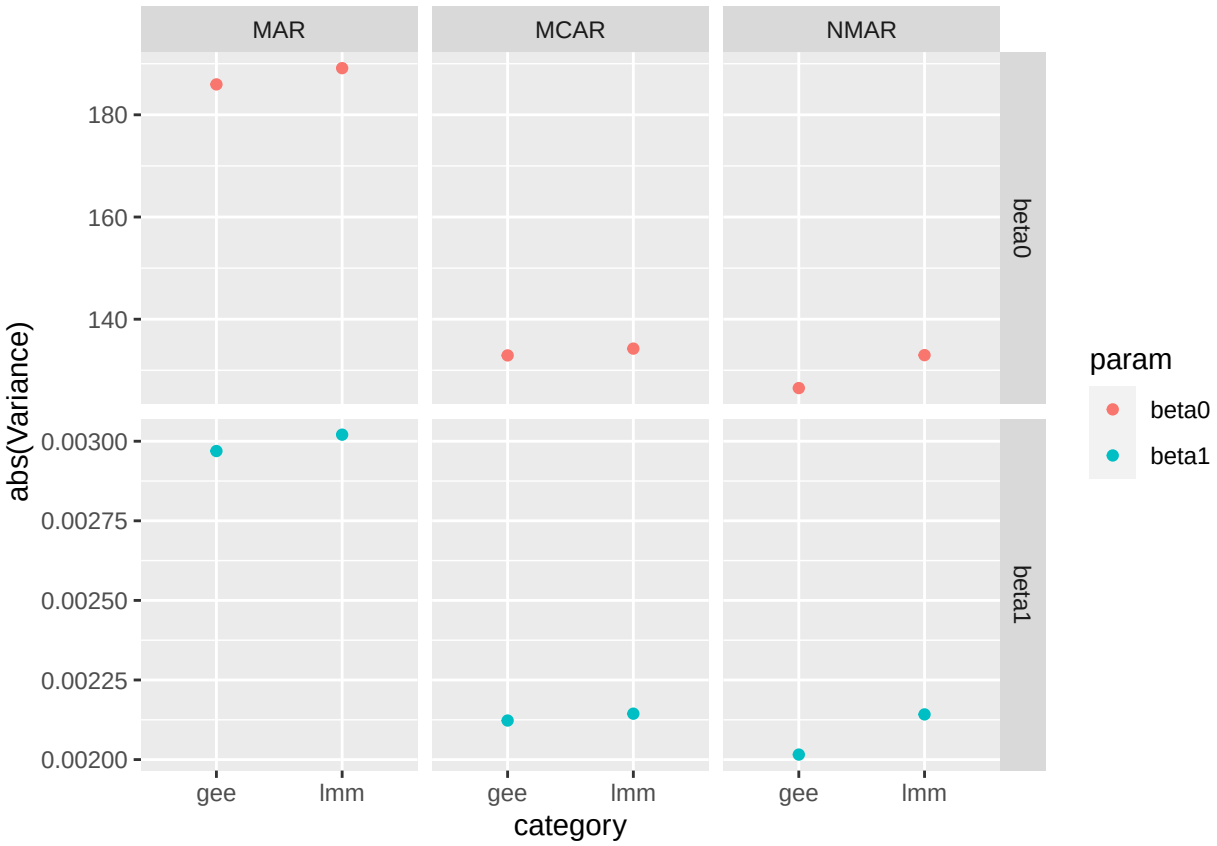
The following code deals with the available case.

```
nrep = 1000

res1 <- do.call(rbind, lapply(c(1:nrep), function(nrep){
  gen.one(ni,nj, case = 2)
}))
mean_res1 <- colMeans(res1)

simulation_res1 = as.data.frame(mean_res1)
simulation_res1
```

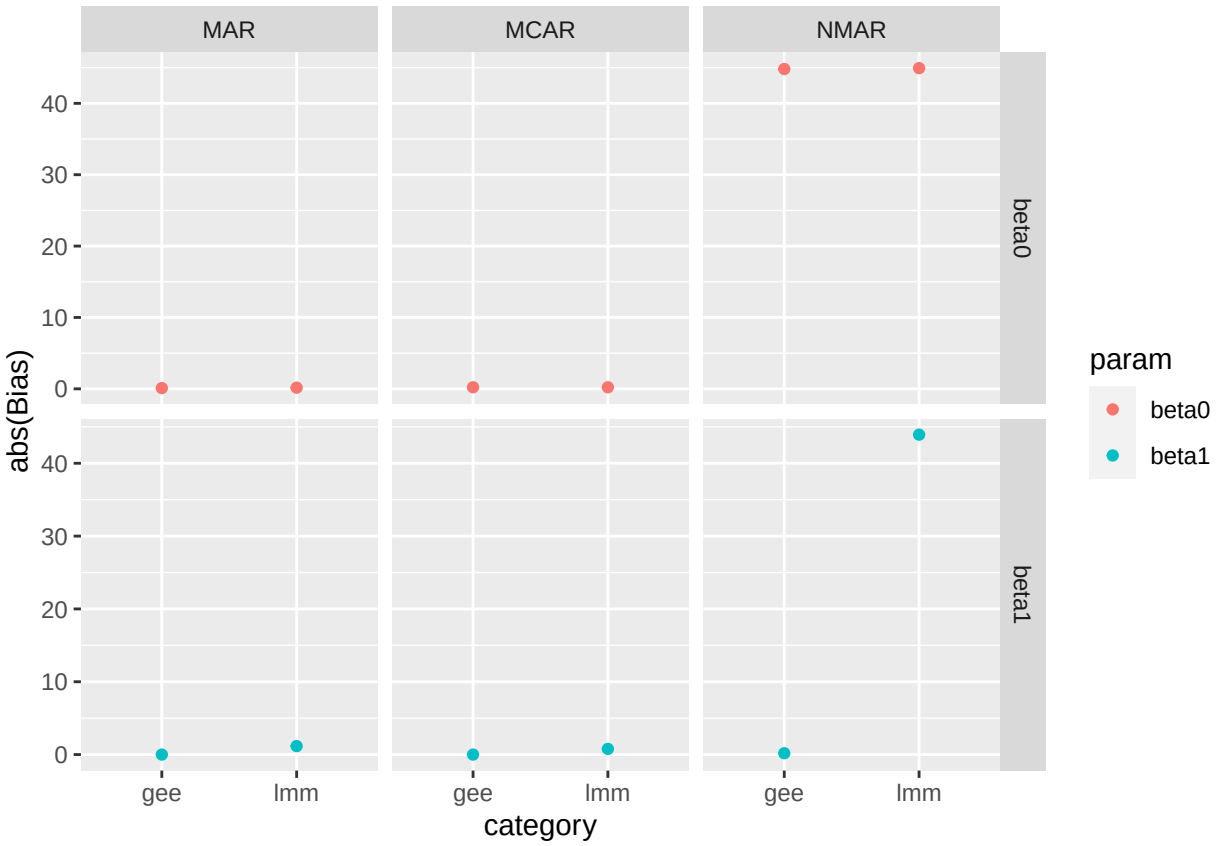
```
##                mean_res1
## lmm1_var0   1.342485e+02
## lmm1_var1   2.144189e-03
## lmm2_var0   1.891318e+02
## lmm2_var1   3.020591e-03
## lmm3_var0   1.329687e+02
## lmm3_var1   2.141951e-03
## gee1_var0   1.329136e+02
## gee1_var1   2.122641e-03
```

```
## gee2_var0   1.859547e+02
## gee2_var1   2.969184e-03
## gee3_var0   1.265246e+02
## gee3_var1   2.015741e-03
## lmm1_bias0  2.200360e-01
## lmm1_bias1 -7.799640e-01
## lmm2_bias0 -1.649842e-01
## lmm2_bias1 -1.164984e+00
## lmm3_bias0  4.492772e+01
## lmm3_bias1  4.392772e+01
## gee1_bias0  2.178882e-01
## gee1_bias1 -8.125502e-04
## gee2_bias0 -1.082741e-01
## gee2_bias1  4.353933e-04
## gee3_bias0  4.481332e+01
## gee3_bias1 -1.854417e-01
```

```
category = c(rep(c("lmm", "gee"), 2, each = 6))
number = c(rep(c("MCAR", "MAR", "NMAR"), 4, each = 2))
metric = c(rep(c("variance", "bias"), each = 12))
param = c(rep(c("beta0", "beta1"),  12))

names(simulation_res1)[1] = "value"
simulation_res1$category = category
simulation_res1$number = number
simulation_res1$metric = metric
simulation_res1$param = as.factor(param)

simulation_res_var1 = simulation_res1[simulation_res1$metric == "variance",]
names(simulation_res_var1)[1] = "Variance"
simulation_res_bias1 = simulation_res1[simulation_res1$metric != "variance",]
names(simulation_res_bias1)[1] = "Bias"

library(ggplot2)
ggplot(data=simulation_res_var1, aes(x=category, y=abs(Variance), color = param))+geom_point()+
  facet_grid(cols=vars(number), vars(param), scales="free")
```




```
ggplot(data=simulation_res_bias1, aes(x=category, y=abs(Bias), color = param))+geom_point()+
  facet_grid(cols=vars(number), vars(param), scales="free")
```



As we can easily observe from the visualization above, the bias shows a trend.NMAR shows high bias for both models in terms of beta0, but gee performs well in beta1. For other cases, lmm might perform slightly worse but in both the MAR and MCAR cases the bias is negligible.

As for variance, MAR has highest variance for both parameters of all cases, followed by MCAR and then NMAR LMM seems always to have a higher variance than GEE in terms of beta0 ande beta1 across all missingness.

The trend in generally similar for both complete and available cases, except for the order of the variance.

Question 2

Statistical Analysis Plan

The statistical analyses to be done include modeling using LMM and GEE methods, using the logitudinal Taylor Swift data, to see if the level of interest and knowledge of her actually makes an impact on the student performance. Note that there might be a lot of other factors to be controlled such as amount of work students put into studying, the difficulty of classes, student financial situation (affording tutors, etc.) and other concerns. But we can make an assumption that these factors won't matter, and Taylor Swift is the sole possible explanatory variable for student performance. In this way, we will use all available data (since some students might drop out of the survey due to graduation, academic leave, drafting, etc.), using the available approach for all data present. After modelling, we can compare the coefficients as well as the standard error, to observe if there is any significant relationship between Taylor Swift and student performance.

Power and sample size

In terms of endpoints, we can use the academic performance with GPA, the easiest metric we can obtain. In terms of mental health, we might need help from mental health professionals in designing some survey to model stress, depression, etc.

The null hypothesis is that there is no significant difference between the groups, and the alternative is Taylor Swift makes students happier and perform better in class. If that's true, we can play Taylor Swift while in class for Stat 571.

The sample size is 200 as specified, 4 observations for each individual makes total observations 800, not considering missing data. The power can be tentatively 90%.

Question 3

Extending Q1, we simply change the data generation function in Q1 to reflect a non-normal (uniform) random effect b.

```
##### generic data setup:

ni     = 100  # 100 subjects
nj     =  10  # 10 observation for each subject
prop.m = .07  # 7% missingness
beta0 = 0.5
beta1 = 1.5

simu_miss1=function(ni,nj){
  x    = rnorm(ni*nj, mean=250, sd=10)
  time   = rep(1:nj, times=ni)
  id     = rep(1:ni, each=nj)

  e = rnorm(ni*nj, mean=0, sd=10)
  b = rep(runif(ni, 0,100),nj)
  y      = beta0 + beta1 * x + b + e

  # MCAR
  mcar   = runif(ni*nj, min=0, max=1)
  y.mcar = ifelse(mcar<prop.m, NA, y)  # unrelated to anything

  # MAR
  y.mar = matrix(y, ncol=nj, nrow=ni, byrow=TRUE)
  for(i in 1:ni){
    for(j in 4:nj){
      dif1 = y.mar[i,j-2]-y.mar[i,j-3]
      dif2 = y.mar[i,j-1]-y.mar[i,j-2]
      if(dif1>0 & dif2>0){  # if weight goes up twice, drops out
        y.mar[i,j:nj] = NA;  break
      }
    }
  }
  y.mar = as.vector(t(y.mar))

  # NMAR
  sort.y = sort(y, decreasing=TRUE)
  nmar   = sort.y[ceiling(prop.m*length(y))]
  y.nmar = ifelse(y>nmar, NA, y)  # doesn't show up when heavier
  return(data.frame(id, time, x, y, y.mcar,y.mar,y.nmar))
}

library(geepack)
library(lme4)
```

```r
gen.one1=function(ni,nj, case = 1){
  df=simu_miss1(ni,nj)
  df1 = df
  df2 = df
  df3 = df

  # complete
  if (case == 1){

    id1 = unique(df[is.na(df$y.mcar), ]$id)
    df1 = df[!(df$id %in% id1), ]

    id2 = unique(df[is.na(df$y.mar), ]$id)
    df2 = df[!(df$id %in% id1), ]

    id3 = unique(df[is.na(df$y.nmar), ]$id)
    df3 = df[!(df$id %in% id1), ]

  }

  # available
  #if (case == 2){
  #  df = na.omit(df)
  #}


  # lmm
  lmm1 = lmer(y.mcar ~ x + (1|id), data = df1, REML = T)
  lmm2 = lmer(y.mar ~ x + (1|id), data = df2, REML = T)
  lmm3 = lmer(y.nmar ~ x + (1|id), data = df3, REML = T)

  # gee
  gee1 = geeglm(y.mcar ~ x, id = id, data = df1, corstr = "exchangeable")
  gee2 = geeglm(y.mar ~ x, id = id, data = df2, corstr = "exchangeable")
  gee3 = geeglm(y.nmar ~ x, id = id, data = df3, corstr = "exchangeable")

  # Estimate variance for efficiency
  lmm1_var0 = vcov(lmm1)[1,1]
  lmm1_var1 = vcov(lmm1)[2,2]
  lmm2_var0 = vcov(lmm2)[1,1]
  lmm2_var1 = vcov(lmm2)[2,2]
  lmm3_var0 = vcov(lmm3)[1,1]
  lmm3_var1 = vcov(lmm3)[2,2]

  gee1_var0 = vcov(gee1)[1,1]
  gee1_var1 = vcov(gee1)[2,2]
  gee2_var0 = vcov(gee2)[1,1]
  gee2_var1 = vcov(gee2)[2,2]
  gee3_var0 = vcov(gee3)[1,1]
  gee3_var1 = vcov(gee3)[2,2]

  # Estimate bias
  lmm1_bias0 = fixef(lmm1)[1] - beta0
```

```
  lmm1_bias1 = fixef(lmm1)[1] - beta1
  lmm2_bias0 = fixef(lmm2)[1] - beta0
  lmm2_bias1 = fixef(lmm2)[1] - beta1
  lmm3_bias0 = fixef(lmm3)[1] - beta0
  lmm3_bias1 = fixef(lmm3)[1] - beta1

  gee1_bias0 = coef(gee1)[1] - beta0
  gee1_bias1 = coef(gee1)[2] - beta1
  gee2_bias0 = coef(gee2)[1] - beta0
  gee2_bias1 = coef(gee2)[2] - beta1
  gee3_bias0 = coef(gee3)[1] - beta0
  gee3_bias1 = coef(gee3)[2] - beta1

  return(data.frame( lmm1_var0, lmm1_var1, lmm2_var0, lmm2_var1,
                     lmm3_var0, lmm3_var1, gee1_var0, gee1_var1,
                     gee2_var0, gee2_var1, gee3_var0, gee3_var1,
                     lmm1_bias0, lmm1_bias1, lmm2_bias0, lmm2_bias1,
                     lmm3_bias0, lmm3_bias1, gee1_bias0, gee1_bias1,
                     gee2_bias0, gee2_bias1, gee3_bias0, gee3_bias1
  ) )
}
```

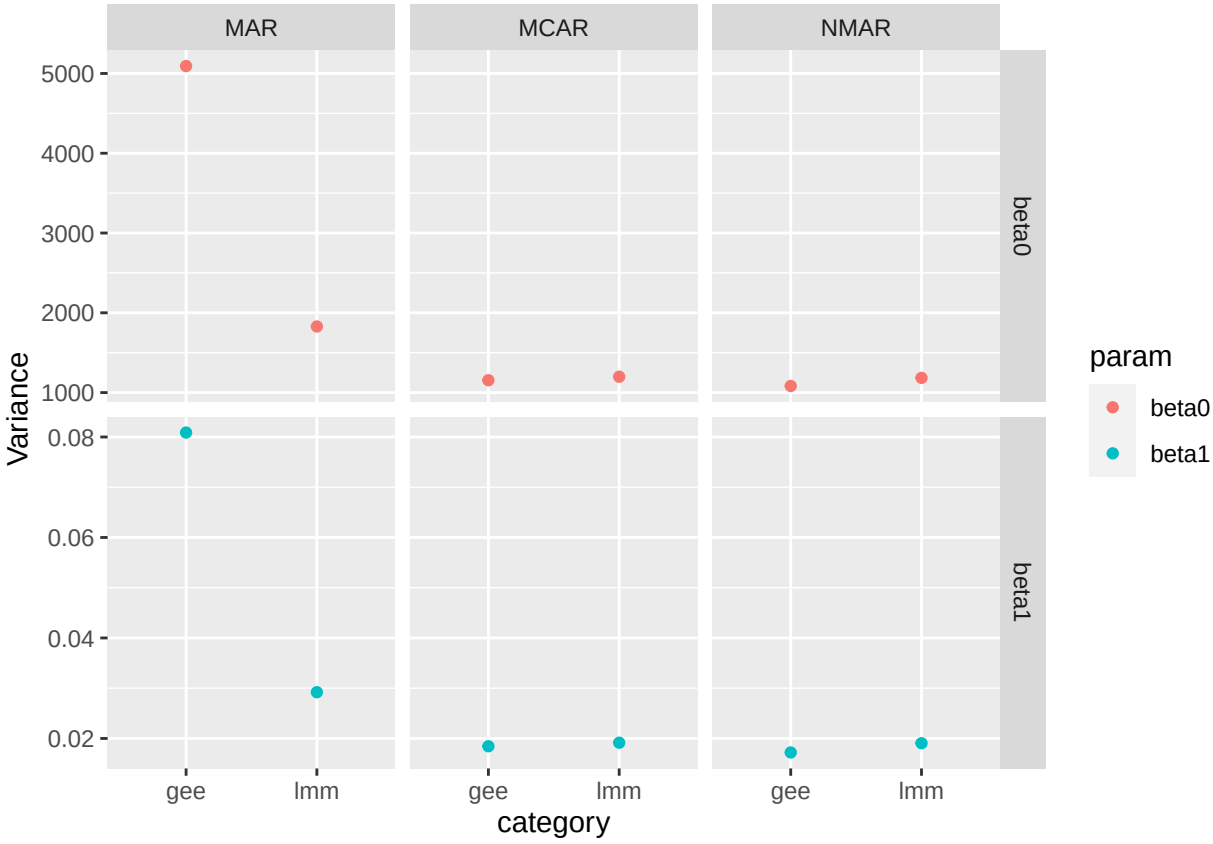
For simplicity, I will only consider the complete case.

```
nrep = 1000

res2 <- do.call(rbind, lapply(c(1:nrep), function(nrep){
  gen.one1(ni,nj, case = 1)
}))
mean_res2 <- colMeans(res2)

simulation_res2 = as.data.frame(mean_res2)
simulation_res2
```

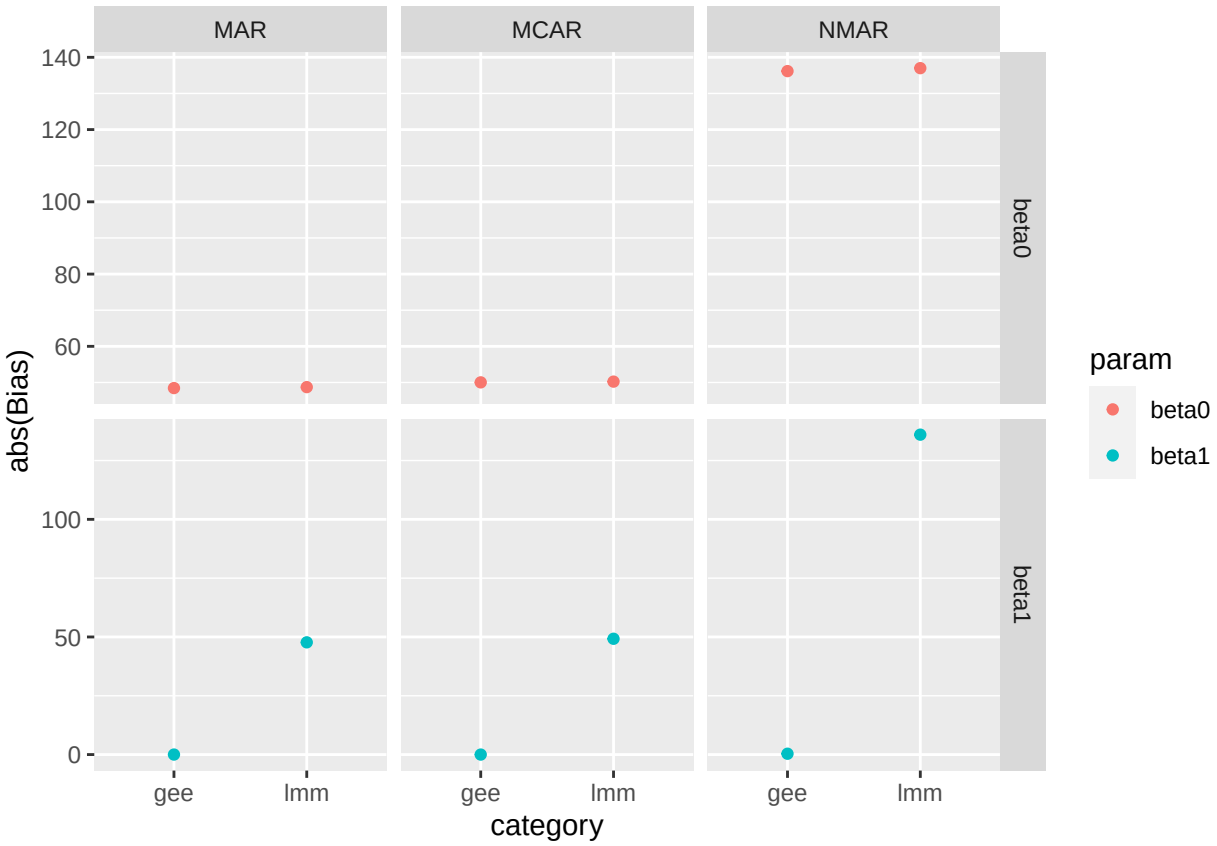
```
##                mean_res2
## lmm1_var0   1.198989e+03
## lmm1_var1   1.914972e-02
## lmm2_var0   1.828324e+03
## lmm2_var1   2.920446e-02
## lmm3_var0   1.185117e+03
## lmm3_var1   1.905044e-02
## gee1_var0   1.154940e+03
## gee1_var1   1.844246e-02
## gee2_var0   5.092968e+03
## gee2_var1   8.088349e-02
## gee3_var0   1.083147e+03
## gee3_var1   1.721052e-02
## lmm1_bias0  5.023297e+01
## lmm1_bias1  4.923297e+01
## lmm2_bias0  4.871265e+01
## lmm2_bias1  4.771265e+01
## lmm3_bias0  1.370001e+02
## lmm3_bias1  1.360001e+02
## gee1_bias0  5.002396e+01
## gee1_bias1 -7.226360e-04
```

```
## gee2_bias0  4.844809e+01
## gee2_bias1  5.492901e-03
## gee3_bias0  1.361825e+02
## gee3_bias1 -3.599601e-01
```

```
category = c(rep(c("lmm", "gee"), 2, each = 6))
number = c(rep(c("MCAR", "MAR", "NMAR"), 4, each = 2))
metric = c(rep(c("variance", "bias"), each = 12))
param = c(rep(c("beta0", "beta1"),  12))

names(simulation_res2)[1] = "value"
simulation_res2$category = category
simulation_res2$number = number
simulation_res2$metric = metric
simulation_res2$param = as.factor(param)

simulation_res_var2 = simulation_res2[simulation_res2$metric == "variance",]
names(simulation_res_var2)[1] = "Variance"
simulation_res_bias2 = simulation_res2[simulation_res2$metric != "variance",]
names(simulation_res_bias2)[1] = "Bias"

library(ggplot2)
ggplot(data=simulation_res_var2, aes(x=category, y=Variance, color = param))+geom_point()+
  facet_grid(cols=vars(number), vars(param), scales="free")
```




```
ggplot(data=simulation_res_bias2, aes(x=category, y=abs(Bias), color = param))+geom_point()+
  facet_grid(cols=vars(number), vars(param), scales="free")
```



Compared with the case where the random effect is normal, the random effect now turns out to make some changes on the output. For example, smaller variance for beta 1 and larger variance for beta 0 in general, similar with bias, although the relative amount between three ways of missing are very similar.