Mengyang Zheng HW 1 3/1/2021

Exercise 1 Missing Data

```
rm(list=ls())
setwd("~/GitHub/Econ613/Assignments/A1/dat")

datstu <- read.csv("~/GitHub/Econ613/Assignments/A1/dat/datstu.csv", stringsAsFactors=TRUE)
datsss <- read.csv("~/GitHub/Econ613/Assignments/A1/dat/datsss.csv", stringsAsFactors=TRUE)
datjss <- read.csv("~/GitHub/Econ613/Assignments/A1/dat/datjss.csv", stringsAsFactors=TRUE)
```

```
#Number of students
nrow(datstu)
```

```
## [1] 340823
```

```
#Number of schools in datstu:640
#Number of shcools in datsss:898
library(tidyr)
length(unique(unlist(na.omit(datstu[5:10][datstu[5:10]!=""]))))
```

```
## [1] 640
```

```
length(unique(datsss[,c("schoolcode")]))
```

```
## [1] 898
```

```
#Number of programs
length(unique(unlist(datstu[11:16][datstu[11:16]!=""])))
```

```
## [1] 32
```

```
#Number of choices
library(stringr)
#If you use the paste command then straight count the unique choices you would get 3086 choices, however, I found this incorrect as paste would make NA real string characters and will generate  cases where if missing schoolcode, then it becomes NA+program or missing program it becomes program+NA, where I think they are invalid choices, so I need to take those choices out.

datstu$choice1=str_c(datstu$schoolcode1," ",datstu$choicepgm1)
datstu$choice2=str_c(datstu$schoolcode2," ",datstu$choicepgm2)
datstu$choice3=str_c(datstu$schoolcode3," ",datstu$choicepgm3)
datstu$choice4=str_c(datstu$schoolcode4," ",datstu$choicepgm4)
datstu$choice5=str_c(datstu$schoolcode5," ",datstu$choicepgm5)
datstu$choice6=str_c(datstu$schoolcode6," ",datstu$choicepgm6)

#Idea is to find choices that have leading blank space or blank space that is in last place of the string after concat 2 columns then it must be an invalid choice.

datstu$STest1=grepl('^ ',datstu$choice1)
datstu$ETest1=grepl(' $',datstu$choice1)
datstu$choice1[datstu$STest1==TRUE | datstu$ETest1==TRUE]<-NA
datstu$STest2=grepl('^ ',datstu$choice2)
datstu$ETest2=grepl(' $',datstu$choice2)
datstu$choice2[datstu$STest2==TRUE | datstu$ETest2==TRUE]<-NA
datstu$STest3=grepl('^ ',datstu$choice3)
datstu$ETest3=grepl(' $',datstu$choice3)
datstu$choice3[datstu$STest3==TRUE | datstu$ETest3==TRUE]<-NA
datstu$STest4=grepl('^ ',datstu$choice4)
datstu$ETest4=grepl(' $',datstu$choice4)
datstu$choice4[datstu$STest4==TRUE | datstu$ETest4==TRUE]<-NA
datstu$STest5=grepl('^ ',datstu$choice5)
datstu$ETest5=grepl(' $',datstu$choice5)
datstu$choice5[datstu$STest5==TRUE | datstu$ETest5==TRUE]<-NA
datstu$STest6=grepl('^ ',datstu$choice6)
datstu$ETest6=grepl(' $',datstu$choice6)
datstu$choice6[datstu$STest6==TRUE | datstu$ETest6==TRUE]<-NA
datstu$STest1=NULL
datstu$ETest1=NULL
datstu$STest2=NULL
datstu$ETest2=NULL
datstu$STest3=NULL
datstu$ETest3=NULL
datstu$STest4=NULL
datstu$ETest4=NULL
datstu$STest5=NULL
datstu$ETest5=NULL
datstu$STest6=NULL
datstu$ETest6=NULL
length(unique(unlist(datstu[19:24]),na.rm=TRUE))
```

```
## [1] 2774
```

```
#Missing test score
sum(is.na(datstu$score))
```

```
## [1] 179887
```

```
#Apply to the same school(different program)
for (i in 1:nrow(datstu)){
datstu$sameschool[i]=length(unique(na.omit(unlist(datstu[i,5:10]))))
}

sum(datstu$sameschool<6-rowSums(is.na(datstu[5:10])))
```

```
## [1] 120071
```

```
datstu$sameschool=NULL
```

```
#Apply to less than 6 choices
datstu$Applynum=6-rowSums(is.na(datstu[19:24]))
sum(datstu$Applynum<6)
```

```
## [1] 21001
```

```
datstu$Applynum=NULL
```

Exercise 2: Data

```
#Reshape using gather
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.0.6     v forcats 0.5.1
## v readr   1.4.0
```

```
## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(reshape2)
```

```
##
## Attaching package: 'reshape2'
```

```
## The following object is masked from 'package:tidyr':
##
##     smiths
```

```
dat_school=dplyr::select(datstu,X,choice1:choice6)
datjss$X = NULL
datsss$X = NULL
dat_school=gather(dat_school,'key','value',-X)

#Merge datstu and datjss
dat_merge=merge(datstu,datjss,by="jssdistrict")

#Merge using X as merge ID
#dat_school is the dataset for all students (admited and non-admited)
dat_school=merge(dat_school,dat_merge,by="X")


#Drop irrelevant variables
dat_school$choice1 = NULL
dat_school$choice2 = NULL
dat_school$choice3 = NULL
dat_school$choice4 = NULL
dat_school$choice5 = NULL
dat_school$choice6 = NULL

dat_school$schoolcode1 = NULL
dat_school$schoolcode2 = NULL
dat_school$schoolcode3 = NULL
dat_school$schoolcode4 = NULL
dat_school$schoolcode5 = NULL
dat_school$schoolcode6 = NULL

dat_school$choicepgm1 = NULL
dat_school$choicepgm2 = NULL
dat_school$choicepgm3 = NULL
dat_school$choicepgm4 = NULL
dat_school$choicepgm5 = NULL
dat_school$choicepgm6 = NULL

#Split the school code and program value column
dat_school=cbind(dat_school,colsplit(dat_school$value," ",c("schoolcode", "program")))


#clean sss data and merge
datsss=datsss[!duplicated(datsss$schoolcode), ]
dat_school=merge(dat_school,datsss,by="schoolcode")

#Keep all choices that students are admitted
dat_school$key[dat_school$key=="choice1"]<-1
dat_school$key[dat_school$key=="choice2"]<-2
dat_school$key[dat_school$key=="choice3"]<-3
dat_school$key[dat_school$key=="choice4"]<-4
dat_school$key[dat_school$key=="choice5"]<-5
dat_school$key[dat_school$key=="choice6"]<-6

names(dat_school)[names(dat_school)=="key"] <- "rankedchoice"
#If choice number meets with the rankplace number, then the student is admitted
```

```
#You can treat dat_school as student choice level data now
dat_admitted=subset(dat_school,rankedchoice==rankplace)

dat_school_level=dat_admitted
#Use group method to find cutoff, quality, and size.
dat_school_level=dat_school_level %>%
                    group_by(value) %>%
                    mutate(cutoff=min(score,na.rm=TRUE))

dat_school_level=dat_school_level %>%
                    group_by(value) %>%
                    mutate(quality=mean(score,na.rm=TRUE))

dat_school_level=dat_school_level %>%
                    group_by(value) %>%
                    mutate(size=n())

dat_school_level=dat_school_level[!duplicated(dat_school_level$value),]
dat_school_level=dat_school_level[c("schoolcode","program","schoolname","cutoff","quality","siz
e","sssdistrict","ssslong","ssslat")]

head(dat_school_level,20)
```

```
## # A tibble: 20 x 9
##    schoolcode program schoolname cutoff quality  size sssdistrict ssslong ssslat
##         <int> <chr>   <fct>       <int>   <dbl> <int> <fct>         <dbl>  <dbl>
##  1      10101 Busine~ EBENEZER ~    305    325.   100 Accra Metr~  -0.197   5.61
##  2      10101 Genera~ EBENEZER ~    316    330.   100 Accra Metr~  -0.197   5.61
##  3      10101 Home E~ EBENEZER ~    284    301.    49 Accra Metr~  -0.197   5.61
##  4      10101 Genera~ EBENEZER ~    299    329.    50 Accra Metr~  -0.197   5.61
##  5      10101 Agricu~ EBENEZER ~    288    310.    49 Accra Metr~  -0.197   5.61
##  6      10101 Visual~ EBENEZER ~    296    312.    50 Accra Metr~  -0.197   5.61
##  7      10102 Genera~ ST. MARY'~    388    405.    88 Accra Metr~  -0.197   5.61
##  8      10102 Home E~ ST. MARY'~    363    377.    45 Accra Metr~  -0.197   5.61
##  9      10102 Visual~ ST. MARY'~    343    371.    45 Accra Metr~  -0.197   5.61
## 10      10102 Genera~ ST. MARY'~    389    406.    70 Accra Metr~  -0.197   5.61
## 11      10103 Genera~ WESLEY GR~    349    363.   117 Accra Metr~  -0.197   5.61
## 12      10103 Agricu~ WESLEY GR~    316    333.    38 Accra Metr~  -0.197   5.61
## 13      10103 Home E~ WESLEY GR~    320    336.    49 Accra Metr~  -0.197   5.61
## 14      10103 Genera~ WESLEY GR~    335    354.    80 Accra Metr~  -0.197   5.61
## 15      10103 Visual~ WESLEY GR~    343    358.    40 Accra Metr~  -0.197   5.61
## 16      10103 Busine~ WESLEY GR~    341    358.   119 Accra Metr~  -0.197   5.61
## 17      10104 Genera~ HOLY TRIN~    302    320.    55 Accra Metr~  -0.197   5.61
## 18      10104 Home E~ HOLY TRIN~    264    286.    55 Accra Metr~  -0.197   5.61
## 19      10104 Visual~ HOLY TRIN~    273    298.    55 Accra Metr~  -0.197   5.61
## 20      10104 Genera~ HOLY TRIN~    245    283.    55 Accra Metr~  -0.197   5.61
```

Exercise 3: Distance

```
#calculate distance between junior high and senior high for all admitted and non-admitted studen
ts and for all of their choices.

dat_school$distance=sqrt((69.172*(dat_school$ssslong-dat_school$point_x)*cos(dat_school$point_y/
57.3))^2+(69.172*(dat_school$ssslat-dat_school$point_y))^2)

#This is for Exercise 4
dat_admitted$distance=sqrt((69.172*(dat_admitted$ssslong-dat_admitted$point_x)*cos(dat_admitted
$point_y/57.3))^2+(69.172*(dat_admitted$ssslat-dat_admitted$point_y))^2)

head(dat_school$distance,20)
```

```
##  [1]  0.00000  0.00000  0.00000  0.00000  0.00000 14.35205 21.87927 14.35205
##  [9]  0.00000  0.00000  0.00000  0.00000  0.00000 21.87927 14.35205 14.35205
## [17] 14.35205 21.87927  0.00000  0.00000
```

Exercise 4: Descriptive Characteristics

```
#merge choice level and school level data then we can compute all ranked choices mean and sd for
relevant variables
dat_school_level_core=dat_school_level[c("schoolcode","program","cutoff","quality")]

dat_choice=merge(dat_school,dat_school_level_core,"value"=paste("schoolcode","program"))

Descriptive=dat_choice %>%
  group_by(rankedchoice) %>%
  summarise(mean_cutoff=mean(cutoff),
       sd_cutoff=sd(cutoff),
       mean_quality=mean(quality),
       sd_quality=sd(quality),
       mean_distance=mean(distance),
            sd_distance=sd(distance))

head(Descriptive)
```

```
## # A tibble: 6 x 7
##   rankedchoice mean_cutoff sd_cutoff mean_quality sd_quality mean_distance
##   <chr>              <dbl>     <dbl>        <dbl>      <dbl>         <dbl>
## 1 1                   294.      54.1         317.       48.5          28.3
## 2 2                   281.      49.6         305.       43.8          28.2
## 3 3                   273.      47.0         297.       41.2          27.3
## 4 4                   263.      45.1         289.       39.3          24.4
## 5 5                   250.      32.1         278.       26.7          28.7
## 6 6                   246.      31.4         274.       26.2          29.5
## # ... with 1 more variable: sd_distance <dbl>
```

```
#Divide each choice into 4 quantiles by student scores
dat_choice$quantiles=ntile(dat_choice$score,4)
Descriptive2=dat_choice %>%
  group_by(rankedchoice,quantiles) %>%
    summarise(mean_cutoff=mean(cutoff),
       sd_cutoff=sd(cutoff),
       mean_quality=mean(quality),
       sd_quality=sd(quality),
       mean_distance=mean(distance),
            sd_distance=sd(distance))
```

```
## `summarise()` has grouped output by 'rankedchoice'. You can override using the `.groups` argu
ment.
```

```
Descriptive2=drop_na(Descriptive2)
head(Descriptive2,24)
```

```
## # A tibble: 24 x 8
## # Groups:   rankedchoice [6]
##    rankedchoice quantiles mean_cutoff sd_cutoff mean_quality sd_quality
##    <chr>            <int>       <dbl>     <dbl>        <dbl>      <dbl>
##  1 1                    1        283.      44.6         307.       38.9
##  2 1                    2        300.      45.1         322.       39.3
##  3 1                    3        322.      43.6         342.       38.4
##  4 1                    4        362.      38.4         380.       35.1
##  5 2                    1        270.      41.4         294.       35.9
##  6 2                    2        285.      42.4         308.       36.6
##  7 2                    3        304.      42.4         325.       36.7
##  8 2                    4        340.      38.5         358.       34.2
##  9 3                    1        261.      40.5         287.       35.0
## 10 3                    2        273.      41.3         298.       35.5
## # ... with 14 more rows, and 2 more variables: mean_distance <dbl>,
## #   sd_distance <dbl>
```

Exercise 5: Data Creation

```
rm(list=ls())
set.seed(0)
X1=runif(10000,min=1,max=3)
X2=rgamma(10000,shape=3,scale=2)
X3=rbinom(10000,size=1,prob=0.3)
E=rnorm(10000,mean=2,sd=1)
Y=0.5+1.2*X1-0.9*X2+0.1*X3+E

ydum=rep(0,10000)

# Loop over 10000 entries, if satisfied, replace 0 with 1
for (j in 1:10000){
  if (Y[j]>mean(Y)){
    ydum[j]=1
  }
}

head(ydum,20)
```

```
##  [1] 1 0 1 0 0 0 1 1 0 0 0 1 0 1 1 0 1 1 1 1
```

Exercise 6: OLS

```
cor(Y, X1)
```

```
## [1] 0.208173
```

```
#Largely different from 1.2, only 0.208
X=as.matrix(cbind(1,X1,X2,X3))
beta=solve(t(X)%*%X)%*%t(X)%*%Y
beta # beta are 1.22,-0.9, and 0.069
```

```
##          [,1]
##     2.4709484
## X1  1.2269162
## X2 -0.9014403
## X3  0.0691434
```

```
SigmaSq=sum((Y-X%*%beta)^2)/(nrow(X)-ncol(X))
Var=SigmaSq*solve(t(X)%*%X)
SE=sqrt(diag(Var))
SE #SE for each beta is 0.0173,0.0029, 0.0219
```

```
##                      X1          X2          X3
## 0.040574670 0.017297362 0.002923809 0.021917400
```

Exercise 7: Discrete Choice

```
#This is the result from package use
X_all=as.data.frame(cbind(ydum,X1,X2,X3))
probit=glm(ydum~X1+X2+X3,data=X_all,family=binomial(link="probit"))
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(probit)
```

```
##
## Call:
## glm(formula = ydum ~ X1 + X2 + X3, family = binomial(link = "probit"),
##     data = X_all)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.5687  -0.1152   0.0092   0.2485   3.4097
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  2.81349    0.09713  28.967  < 2e-16 ***
## X1           1.26453    0.04359  29.011  < 2e-16 ***
## X2          -0.88993    0.01811 -49.141  < 2e-16 ***
## X3           0.12688    0.04688   2.706  0.00681 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 13700.7  on 9999  degrees of freedom
## Residual deviance:  4351.5  on 9996  degrees of freedom
## AIC: 4359.5
##
## Number of Fisher Scoring iterations: 7
```

```
#Now we compute using optim function
probit_log<-function(x,y,beta){
  prob <- pnorm(x %*% beta)
  -sum((1-y)*log(1-prob)+y*log(prob))
}

probit_gr <- function(x,y,beta){
  prob <- pnorm(x %*% beta)
  grad <- dnorm(x %*% beta)*(y-prob)/(prob*(1-prob))
  -crossprod(x,grad)
}


X0=as.matrix(cbind(1,X1,X2,X3))
Y0=as.matrix(ydum)
probit0 <- optim(par=c(0.1,0.1,0.1,0.1),probit_log,y=Y0,x=X0,gr=probit_gr, method="BFGS", hessia
n=TRUE)

probit0$par
```

```
## [1]  2.8134891  1.2645285 -0.8899281  0.1268804
```

```
#Same as the one we use the package to solve for, which means this is correct.
```

```
#Package result for logit model
logit=glm(ydum~X1+X2+X3,data=X_all,family=binomial(link="logit"))
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(logit)
```

```
##
## Call:
## glm(formula = ydum ~ X1 + X2 + X3, family = binomial(link = "logit"),
##     data = X_all)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.2457  -0.1513   0.0412   0.2598   3.1498
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  5.05546    0.18151  27.852  < 2e-16 ***
## X1           2.27775    0.08157  27.924  < 2e-16 ***
## X2          -1.60180    0.03625 -44.190  < 2e-16 ***
## X3           0.23182    0.08451   2.743  0.00608 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 13700.7  on 9999  degrees of freedom
## Residual deviance:  4361.6  on 9996  degrees of freedom
## AIC: 4369.6
##
## Number of Fisher Scoring iterations: 7
```

```
#Computation by optim
logit_exp<-function(x,y,beta){
  -sum(y*(x %*% beta - log(1+exp(x %*% beta)))
    + (1-y)*(-log(1 + exp(x %*% beta))))
}

logit0 <- optim(par=c(0.1,0.1,0.1,0.1),logit_exp,y=Y0,x=X0, method="BFGS", hessian=TRUE)
logit0$par
```

```
## [1]  5.0554578  2.2777408 -1.6017983  0.2318224
```

```
#Same as the result coming from the package so it must be correct.
```

```
linear=lm(ydum~X1+X2+X3,data=X_all)
summary(linear)
```

```
##
## Call:
## lm(formula = ydum ~ X1 + X2 + X3, data = X_all)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.90486 -0.26831  0.05573  0.24897  1.77443
##
## Coefficients:
##               Estimate Std. Error  t value Pr(>|t|)
## (Intercept)  0.8717711  0.0133173   65.462   <2e-16 ***
## X1           0.1570332  0.0056773   27.660   <2e-16 ***
## X2          -0.1042533  0.0009596 -108.638   <2e-16 ***
## X3           0.0125766  0.0071936    1.748   0.0804 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3304 on 9996 degrees of freedom
## Multiple R-squared:  0.5564, Adjusted R-squared:  0.5562
## F-statistic:  4178 on 3 and 9996 DF,  p-value: < 2.2e-16
```

#The sign of coefficients of X1,X2, and X3 are consistent across three models. X1 and X3 turns out to be positive and X2 is negative. However, the maginitude of the probit and the linear model coefficients are relatively close to each other but the coefficients of the logit model are almost twice of the above two models. Coefficients of X1 and X2 are both significant across probit, logit, and linear models. However, the coefficient of X3 is insignificant in linear model but significant in both probit and logit models.

Exercise 8: Marginal Effects

```
library(mfx)
```

```
## Loading required package: sandwich
```

```
## Loading required package: lmtest
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
## Loading required package: MASS
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
## Loading required package: betareg
```

```
#Use Package to get numbers but I have the code for how to get the results, I just want to make
 sure they show the exact same results.
probitmfx(ydum~X1+X2+X3,data=X_all,atmean=FALSE)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Call:
## probitmfx(formula = ydum ~ X1 + X2 + X3, data = X_all, atmean = FALSE)
##
## Marginal Effects:
##          dF/dx   Std. Err.         z     P>|z|
## X1  0.15277673  0.00429272   35.5898 < 2.2e-16 ***
## X2 -0.10751855  0.00040946 -262.5845 < 2.2e-16 ***
## X3  0.01527217  0.00561433    2.7202  0.006524 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## dF/dx is for discrete change for the following variables:
##
## [1] "X3"
```

```
#Use Package to get numbers so we can double check later
logitmfx(ydum~X1+X2+X3,data=X_all,atmean=FALSE)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Call:
## logitmfx(formula = ydum ~ X1 + X2 + X3, data = X_all, atmean = FALSE)
##
## Marginal Effects:
##         dF/dx  Std. Err.        z     P>|z|
## X1  0.1526834  0.0078881  19.3562 < 2.2e-16 ***
## X2 -0.1073730  0.0046848 -22.9194 < 2.2e-16 ***
## X3  0.0154818  0.0056123   2.7586  0.005806 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## dF/dx is for discrete change for the following variables:
##
## [1] "X3"
```

```
#Use class Code to compute the probit marginal effect, same as package ones
#Recall X0=as.matrix(cbind(1,X1,X2,X3))
marg_probit=as.matrix(mean(dnorm(X0 %*% probit0$par))*probit0$par)
marg_probit
```

```
##             [,1]
## [1,]  0.33991761
## [2,]  0.15277668
## [3,] -0.10751854
## [4,]  0.01532932
```

```
#Use class Code to compute the logit marginal effect, same as package ones
marg1_logit=as.matrix(mean(dnorm(X0 %*% logit0$par))*logit0$par)
marg1_logit
```

```
##             [,1]
## [1,]  0.34460242
## [2,]  0.15526091
## [3,] -0.10918568
## [4,]  0.01580204
```

```
#Use library code from mfx to compute the probit SE
xm=as.matrix(colMeans(X_all))
be=as.matrix(probit0$par)
k1=length(probit0$par)
xb=t(xm) %*% be
vcv=solve(probit0$hessian)
gr = apply(cbind(1,X1,X2,X3), 1, function(x){
  as.numeric(as.numeric(dnorm(x %*% be))*(diag(k1) - as.numeric(x %*% be)*(be %*% t(x))))
    })
gr = matrix(apply(gr,1,mean),nrow=k1)

SE_probit_marg = sqrt(diag(gr %*% vcv %*% t(gr)))
SE_probit_marg
```

```
## [1] 0.0096079814 0.0043126089 0.0004104417 0.0056524276
```

```
#Same as package results
```

```
#Use library code from mfx to compute the logit SE
xm=as.matrix(colMeans(X_all))
be=as.matrix(logit0$par)
k1=length(logit0$par)
xb=t(xm) %*% be
vcv=solve(logit0$hessian)
gr = apply(cbind(1,X1,X2,X3), 1, function(x){
      as.numeric(as.numeric(plogis(x %*% be)*(1-plogis(x %*% be)))*
                   (diag(k1) - (1 - 2*as.numeric(plogis(x %*% be)))*(be %*% t(x))))
    })
    gr = matrix(apply(gr,1,mean),nrow=k1)
SE_logit_marg = sqrt(diag(gr %*% vcv %*% t(gr)))

SE_logit_marg
```

```
## [1] 0.017564968 0.007888292 0.004684953 0.005694274
```

```
#Same as package results
```