A3

Yuan Tien

3/3/2022

```
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(tidyr)
library(data.table)
```

```
##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```
getwd()
```

```
## [1] "/Users/yuantien/Desktop/R/613/Data"
```

```
setwd("/Users/yuantien/Desktop/R/613/Data")
list.files()
```

```
##  [1] "A1.html"          "A1.R"             "A2.R"             "A2Note.nb.html"  
##  [5] "A2Note.Rmd"       "A3.Rmd"           "dat_choices.dta"  "dathh2004.csv"   
##  [9] "dathh2005.csv"    "dathh2006.csv"    "dathh2007.csv"    "dathh2008.csv"   
## [13] "dathh2009.csv"    "dathh2010.csv"    "dathh2011.csv"    "dathh2012.csv"   
## [17] "dathh2013.csv"    "dathh2014.csv"    "dathh2015.csv"    "dathh2016.csv"   
## [21] "dathh2017.csv"    "dathh2018.csv"    "dathh2019.csv"    "datind2004.csv"  
## [25] "datind2005.csv"   "datind2006.csv"   "datind2007.csv"   "datind2008.csv"  
## [29] "datind2009.csv"   "datind2010.csv"   "datind2011.csv"   "datind2012.csv"  
## [33] "datind2013.csv"   "datind2014.csv"   "datind2015.csv"   "datind2016.csv"  
## [37] "datind2017.csv"   "datind2018.csv"   "datind2019.csv"   "datjss.csv"      
## [41] "datsss.csv"       "datstu_v2.csv"    "game1.R"          "logti_data.RData"
## [45] "param.RData"      "try5.csv"
```

```
datstu <- fread("datstu_v2.csv")
datsch <- fread("datsss.csv")
geo    <- fread("datjss.csv")
datsss <- fread("datsss.csv")
```

Exercise 1
1.1

```
programs <- datstu[,11:16] #select all progran choices from the dataset
programs <- unlist(programs, use.names = FALSE) #turn programs from choice 1 to choic
e 6 into a vector

uniprog <- unique(programs)

Num <- data.frame(no._student = nrow(datstu),
                  no._school  = length(unique(datsch$schoolcode)),
                  no._program = length(uniprog) )
Num
```

```
##   no._student no._school no._program
## 1      340823        898          33
```

1.2 unique school - program dyads Can I just paste school choice with corresponding dyads and find the unique ones?

```
matchoice1 <- datstu%>%
  select(schoolcode1, choicepgm1)

matchoice2 <- datstu%>%
  select(schoolcode2, choicepgm2)

matchoice3 <- datstu%>%
  select(schoolcode3, choicepgm3)

matchoice4 <- datstu%>%
  select(schoolcode4, choicepgm4)

matchoice5 <- datstu%>%
  select(schoolcode5, choicepgm5)

matchoice6 <- datstu%>%
  select(schoolcode6, choicepgm6)

#apropos()

allchoice <- do.call("rbind", list(matchoice1, matchoice2, matchoice3, matchoice4, ma
tchoice5, matchoice6, use.names=FALSE))

choice <- unique(allchoice)
nrow(choice) #3086 unique school - programs dyads
```

```
## [1] 3086
```

1.3 apply to schools near home

```
schdis <- datsch %>%
  select(schoolcode, sssdistrict)
schdis <- schdis[!duplicated(schdis$schoolcode),] #This is a list of schools with corresponding district

library(dplyr)

x <- datstu
schdis <- rename(schdis, schoolcode1 = schoolcode)
  x <- x %>%
    left_join(schdis, by = "schoolcode1")
  schdis <- rename(schdis, schoolcode2 = schoolcode1)
  x <- x %>%
    left_join(schdis, by = "schoolcode2")
  schdis <- rename(schdis, schoolcode3 = schoolcode2)
  x <- x %>%
    left_join(schdis, by = "schoolcode3")
  schdis <- rename(schdis, schoolcode4 = schoolcode3)
  x <- x %>%
    left_join(schdis, by = "schoolcode4")
  schdis <- rename(schdis, schoolcode5 = schoolcode4)
  x <- x %>%
    left_join(schdis, by = "schoolcode5")
  schdis <- rename(schdis, schoolcode6 = schoolcode5)
  x <- x %>%
    left_join(schdis, by = "schoolcode6")
  #by this I create a lot of columns with district name. Then I will determine if jss district equals these columns

  schdis <- rename(schdis, schoolcode = schoolcode6)

x$applyhome_1 <- ifelse(x[,17] == x [,19], 1, 0)
x$applyhome_2 <- ifelse(x[,17] == x [,20], 1, 0)
x$applyhome_3 <- ifelse(x[,17] == x [,21], 1, 0)
x$applyhome_4 <- ifelse(x[,17] == x [,22], 1, 0)
x$applyhome_5 <- ifelse(x[,17] == x [,23], 1, 0)
x$applyhome_6 <- ifelse(x[,17] == x [,24], 1, 0)

x <- x %>%
  mutate(applyhome_total = applyhome_1 + applyhome_2 +applyhome_3 + applyhome_4 + applyhome_5 + applyhome_6) %>%
  count(applyhome_total)
applyhome_at_least_one <- sum( x[2:7,2] )
applyhome_at_least_one #250806
```

```
## [1] 250806
```

1.4

```
#I calculate the number of schools admitted students by their rank. For example, to calculate how many students Duke admitted, I add up the number of students get admitted when Duke is their 1st - 6th choice.

try<- datstu %>%
  select(c(5:10,18))

admit_fun <- function (try) {
try$admit_1 <- ifelse(try$rankplace == 1, try$schoolcode1, 0)
ch1 <- count(try, admin = admit_1) #first choices

try$admit_2 <- ifelse(try$rankplace == 2, try$schoolcode1, 0)
ch2 <- count(try, admin = admit_2)

try$admit_3 <- ifelse(try$rankplace == 3, try$schoolcode1, 0)
ch3 <- count(try, admin = admit_3)

try$admit_4 <- ifelse(try$rankplace == 4, try$schoolcode1, 0)
ch4 <- count(try, admin = admit_4)

try$admit_5 <- ifelse(try$rankplace == 5, try$schoolcode1, 0)
ch5 <- count(try, admin = admit_5)

try$admit_6 <- ifelse(try$rankplace == 6, try$schoolcode1, 0)
ch6 <- count(try, admin = admit_6)

school_admin <- bind_rows(ch1, ch2, ch3, ch4, ch5, ch6) %>%
  group_by(admin) %>%
  summarise( sum(n) )

colnames(school_admin) <- c("schoolcode", "admitted number of students")

schoolist <- datsch %>%
  filter(duplicated(schoolcode) == FALSE) %>%
  select(schoolcode, schoolname) %>%
  left_join(school_admin, by = "schoolcode")

return(schoolist)
}

admit_fun(try)
```

```
##      schoolcode                                   schoolname
##   1:      30107       WESLEY GIRLS HIGH SCHOOL, CAPE COAST
##   2:      30103  HOLY CHILD SENIOR HIGH SCHOOL, CAPE COAST
##   3:      21003 ST. PETER'S SENIOR HIGH SCH, NKWATIA-KWAHU
##   4:      10111     PRESBY BOYS SENIOR HIGH. SCHOOL, LEGON
##   5:      30104  MFANTSIPIM SENIOR HIGH SCHOOL, CAPE COAST
##  ---
## 894:      60306
## 895:      10169
## 896:      71107           TAPAMAN SENIOR SENIOR HIGH SCH
## 897:      30204             ST THERESA'S SEMINARY, AMISANO
## 898:      80703
##      admitted number of students
##   1:                        1462
##   2:                        1107
##   3:                        1112
##   4:                        1471
##   5:                        1074
##  ---
## 894:                          NA
## 895:                          NA
## 896:                          NA
## 897:                          NA
## 898:                          NA
```

1.5 To calculate the cutoff of each senior high schools and later on the quality of senior high, I will first create a column showing the student’s admitted school.

I use my “try” dataframe I created earlier to do this. The try dataframe has six separate columns (for 6 school choices) showing each student’s admitted school’s school code. If Mike is admitted to his 3rd dream school of schoolcode 98021, the admit_3 column will show “98021” while admit_1, admit_2….will show 0.

```
adschool_fun <- function (try) {
try$admit_1 <- ifelse(try$rankplace == 1, try$schoolcode1, 0)
try$admit_2 <- ifelse(try$rankplace == 2, try$schoolcode1, 0)
try$admit_3 <- ifelse(try$rankplace == 3, try$schoolcode1, 0)
try$admit_4 <- ifelse(try$rankplace == 4, try$schoolcode1, 0)
try$admit_5 <- ifelse(try$rankplace == 5, try$schoolcode1, 0)
try$admit_6 <- ifelse(try$rankplace == 6, try$schoolcode1, 0)
return(try)
}
try <- adschool_fun(try)

admit <- try[,8:13] #select admit_1 to admit_6
admit$admit_school = rowSums(admit)
#Since columns outside of student's admitted school's rank will show 0, by adding all
the columns I get the school they are admitted to

datstu_ad <- cbind(datstu, admit_school = admit$admit_school)

datstu_ad %>%
  group_by(admit_school) %>%
  summarise(min(score))  #Note that school "0" indicates a pool of people who didn't
 get admitted to any senior high schools
```

```
## # A tibble: 574 × 2
##    admit_school `min(score)`
##           <dbl>        <int>
##  1            0          192
##  2        10101          213
##  3        10102          226
##  4        10103          214
##  5        10104          218
##  6        10105          205
##  7        10106          216
##  8        10107          209
##  9        10108          207
## 10        10109          194
## # … with 564 more rows
```

1.6

```
datstu_ad %>%
  group_by(admit_school) %>%
  summarise( mean(score) )
```

```
## # A tibble: 574 × 2
##    admit_school `mean(score)`
##           <dbl>         <dbl>
##  1            0          259.
##  2        10101          287.
##  3        10102          351.
##  4        10103          306.
##  5        10104          282.
##  6        10105          325.
##  7        10106          302.
##  8        10107          278.
##  9        10108          274.
## 10        10109          271.
## # … with 564 more rows
```

Exercise 2 - Data

During 1.2, I have already compiled a school-program level dataset named “choice”. I will continue to use this.

```
choice <- rename(choice, schoolcode = schoolcode1, program = choicepgm1)

#Since the professor may want school-program level answer, I have a school-program ve
rsion:

#Calculate the school program level
try2 <- datstu %>%
  mutate(schpro1 = paste0(schoolcode1, choicepgm1), schpro2 = paste0(schoolcode2, cho
icepgm2),
         schpro3 = paste0(schoolcode3, choicepgm3), schpro4 = paste0(schoolcode4, cho
icepgm4),
         schpro5 = paste0(schoolcode5, choicepgm5), schpro6 = paste0(schoolcode6, cho
icepgm6)) %>%
  select(2:4, 18:24) #just select useful column

try2$admit1 <- ifelse(try2$rankplace == 1, try2$schpro1, NA)

try2$admit2 <- ifelse(try2$rankplace == 2, try2$schpro2, NA)

try2$admit3 <- ifelse(try2$rankplace == 3, try2$schpro3, NA)

try2$admit4 <- ifelse(try2$rankplace == 4, try2$schpro4, NA)

try2$admit5 <- ifelse(try2$rankplace == 5, try2$schpro5, NA)

try2$admit6 <- ifelse(try2$rankplace == 6, try2$schpro6, NA)

try2 <- try2 %>%
  unite("admit", admit1, admit2, admit3, admit4, admit5, admit6, na.rm=TRUE, remove =
FALSE) #use unite to past multiple columns

schpro_admit <- as.data.frame( table(try2$admit) )
schpro_admit[1,1] <- as.factor("no school or program")
```

```
## Warning in `[<-.factor`(`*tmp*`, iseq, value = structure(1L, .Label = "no school
## or program", class = "factor")): invalid factor level, NA generated
```

```
colnames(schpro_admit) <- c("admit", "count")
schpro_admit
```

```
##                                      admit  count
## 1                                     <NA> 201599
## 2                       100101General Arts     79
## 3                     100101Home Economics     40
## 4                          100101Technical     49
## 5                        100102Agriculture     90
## 6                           100102Business     90
## 7                       100102General Arts     90
## 8                    100102General Science     90
## 9                     100102Home Economics     45
## 10                       100102Visual Arts     45
## 11                      100104General Arts     45
## 12                   100104General Science     45
## 13                    100104Home Economics     45
## 14                          100105Business     80
## 15                      100105General Arts     80
## 16                    100105Home Economics     80
## 17                       100106Agriculture     40
## 18                          100106Business     40
## 19                      100106General Arts     40
## 20                          100201Business     80
## 21                      100201General Arts     40
## 22                   100201General Science     80
## 23                          100202Business    200
## 24                      100202General Arts    250
## 25                   100202General Science    100
## 26                       100203Agriculture     50
## 27                    100203Home Economics     36
## 28                         100203Technical     16
## 29                          100204Business     32
## 30                      100204General Arts     40
## 31                    100204Home Economics     17
## 32                       100301Agriculture     52
## 33                      100301General Arts     34
## 34                    100301Home Economics     16
## 35                          100302Business     50
## 36                      100302General Arts     90
## 37                   100302General Science     45
## 38                    100302Home Economics     50
## 39                       100303Agriculture     45
## 40                    100303Home Economics     22
## 41                          100304Business     50
## 42                    100304Home Economics     50
## 43                       100304Visual Arts     36
## 44                          100401Business     40
## 45                      100401General Arts    120
## 46                   100401General Science     80
## 47                    100401Home Economics     40
## 48                       100401Visual Arts     40
## 49                       100402Agriculture     50
## 50                    100402Home Economics     46
## 51                         100402Technical     14
## 52                       100501Agriculture     90
## 53                    100501Home Economics     90
## 54                         100501Technical     45
## 2295         9090401Welding & Fabrication      6
## 2296     9100101Block Laying & Concreting      7
## 2297           9100101Carpentry & Joinery      5
## 2298 9100101Electrical Installation Works     23
## 2299                9100101Fashion Design      4
## 2300       9100101Mech. Eng. Craft Pract.      5
## 2301           9100101Motor Vehicle Mech.      2
```

```
#this shows how many people are admitted to each school - program
```

```
schpro_cutqua <- try2 %>%
  group_by(admit) %>%
  summarise(cutoff = min(score), quality = mean(score) )

schpro_admit <- schpro_admit %>%
  full_join(schpro_cutqua, by = "admit")

choice2 <- choice %>%
  mutate(admit = paste0(schoolcode, program))

SP <- choice2 %>%
  left_join(schdis, by = "schoolcode") %>%
  left_join(datsss, by = "sssdistrict") %>%
  left_join(schpro_admit, by = "admit")

SP <- rename(SP, sch_n_pgm = admit)

#This SP dataset contains cutoff, quality, and size of school-program. If a school-pr
ogram has NA in cutoff, quality, or size, it means no student is admitted.
```

Exercise 3 Distance

I already compile a “datstu_ad” dataframe that contains the school each student gets admitted to.

```
datstu_ad <- rename(datstu_ad, schoolcode = admit_school)

jss <- fread("datjss.csv") #I am reloading thess again to make sure I didn't change s
th.
sss <- fread("datsss.csv")

sss <- sss[!duplicated(sss$schoolcode),] #filter out duplicate rows

jss <- jss %>%
  rename(jsslong = point_x, jsslat = point_y)

dis_stu <- datstu_ad %>%
  left_join(sss, by = "schoolcode") %>%      #information on admitted senior high sch
ool
  left_join(jss, by = "jssdistrict") %>%     #info on junior high school
  select(ssslong, jsslong, jsslat, ssslat) #select useful columns

dis_stu <- dis_stu %>%
  mutate(dist = sqrt( (69.172*(ssslong - jsslong) * cos(jsslat/57.3)) ^2  +  (69.172
 * (ssslat - jsslat))^2 ))

#the dist column shows the computed distance
```

Exercise 4

```r
try3 <- datstu
try3$scode_rev1 <- substr(try3$schoolcode1, 1, 3)
try3$scode_rev2 <- substr(try3$schoolcode2, 1, 3)
try3$scode_rev3 <- substr(try3$schoolcode3, 1, 3)
try3$scode_rev4 <- substr(try3$schoolcode4, 1, 3)
try3$scode_rev5 <- substr(try3$schoolcode5, 1, 3)
try3$scode_rev6 <- substr(try3$schoolcode6, 1, 3)

#I initially want to do it in a pipeline but it returns "unused argument" all the time

arts <- c("General Arts", "Visual Arts")
economics <- c("Business", "Home Economics")
science <- "General Science"

try3 <- within(try3, {
  pgm_rev1 = "others"
  pgm_rev1[choicepgm1 %in% arts] = "arts"
  pgm_rev1[choicepgm1 %in% economics] = "economics"
  pgm_rev1[choicepgm1 %in% science] = "science"
  pgm_rev1[is.na(pgm_rev1) == T] = "others"

  pgm_rev2 = "others"
  pgm_rev2[choicepgm2 %in% arts] = "arts"
  pgm_rev2[choicepgm2 %in% economics] = "economics"
  pgm_rev2[choicepgm2 %in% science] = "science"
  pgm_rev2[is.na(pgm_rev2) == T] = "others"

  pgm_rev3 = "others"
  pgm_rev3[choicepgm3 %in% arts] = "arts"
  pgm_rev3[choicepgm3 %in% economics] = "economics"
  pgm_rev3[choicepgm3 %in% science] = "science"
  pgm_rev3[is.na(pgm_rev3) == T] = "others"

  pgm_rev4 = "others"
  pgm_rev4[choicepgm4 %in% arts] = "arts"
  pgm_rev4[choicepgm4 %in% economics] = "economics"
  pgm_rev4[choicepgm4 %in% science] = "science"
  pgm_rev4[is.na(pgm_rev4) == T] = "others"

  pgm_rev5 = "others"
  pgm_rev5[choicepgm5 %in% arts] = "arts"
  pgm_rev5[choicepgm5 %in% economics] = "economics"
  pgm_rev5[choicepgm5 %in% science] = "science"
  pgm_rev5[is.na(pgm_rev5) == T] = "others"

  pgm_rev6 = "others"
  pgm_rev6[choicepgm6 %in% arts] = "arts"
  pgm_rev6[choicepgm6 %in% economics] = "economics"
  pgm_rev6[choicepgm6 %in% science] = "science"
  pgm_rev6[is.na(pgm_rev6) == T] = "others"
})

#Be caution that if a student does not submit a choice (choice = NA) , it will be considered "others"
```

Choice variable

```
try3 <- try3 %>%
  mutate(choice_rev1 = paste0(scode_rev1, pgm_rev1), choice_rev2 = paste0(scode_rev2,
pgm_rev2),
         choice_rev3 = paste0(scode_rev3, pgm_rev3), choice_rev4 = paste0(scode_rev4,
pgm_rev4),
         choice_rev5 = paste0(scode_rev5, pgm_rev5), choice_rev6 = paste0(scode_rev6,
pgm_rev6))
```

Compute new quality and cutoff

```
cutqua <- function(x) {
x$admit1 <- ifelse(x$rankplace == 1, x$choice_rev1, NA)
x$admit2 <- ifelse(x$rankplace == 2, x$choice_rev2, NA)
x$admit3 <- ifelse(x$rankplace == 3, x$choice_rev3, NA)
x$admit4 <- ifelse(x$rankplace == 4, x$choice_rev4, NA)
x$admit5 <- ifelse(x$rankplace == 5, x$choice_rev5, NA)
x$admit6 <- ifelse(x$rankplace == 6, x$choice_rev6, NA)

x <- x %>%
  unite("admit", admit1, admit2, admit3, admit4, admit5, admit6, na.rm=TRUE, remove =
FALSE)

x %>%
  group_by(admit) %>%
  summarise(cutoff = min(score), quality = mean(score) )

}

new_cutqua <- cutqua(try3) #This will show the cutoff and quality of each newly compi
led school - program category
```

Consider the 20,000 highest score students

```
try4 <- try3[order(-score), ] #negative sign means descending
try4 <- try4[1:20000,]
```

Exercise 5

Note that the first choice is choice_rev1 First choice is a catego

```
length(unique(try4$choice_rev1))
```

```
## [1] 246
```

```
# Dependent Variable: choice_rev1, categorical, 246 choices
# Independent Variable: test score, continuous
# Since we are dealing with student characteristic and their preference of school-program, we should use multinomial logit.

try5<- try4

try5$choice_rev1 <- as.numeric( as.factor(try5$choice_rev1) )

name_list <- try4 %>%       #this list stores the factor number and corresponding school-pgm name
  select(choice_rev1) %>%
  cbind(try5$choice_rev1)

like_fun1 <- function(par, try5) {
  choice_rev1 = try5$choice_rev1
  score = try5$score

  n_i = nrow(try5) #should be 20,000 students
  n_j = length(unique(try5$choice_rev1)) #246 choices
  out = mat.or.vec( n_i,n_j )
  #This out should eventually contain the imagined utility for every individual and their potential choice

  #remember to omit a choice as the reference choice
  n_jref = n_j - 1

  #Since restrict Beta_omitted_choice = 0 means the choice essentially has no effect on utility, I can set the utility of that choice to 0 to represent restriction
  out[,1] = 0

  #parameter set for every right-hand side variables and intercept
  par_set1 = par[1:n_jref]
  par_set2 = par[ (n_jref+1) : (2*n_jref) ]

  for (j in 2:n_j) { #remember out[,1] should be 0, so we should start from the second column
    out[,j] = par_set1[(j-1)] + par_set2[j-1] * score
  }

  #transform the utility to form logit probility
  prob = exp(out)
  prob = sweep(prob, MARGIN=1, FUN="/", STATS=rowSums(prob))
  #margin = 1 means operate by row. This sweeps function means we do every exp(XiBj)/(exp(XiBj) + exp(XiBe) +exp(XiBk)...) by row

  prob_choice = NULL
for (i in 1:n_i){
    prob_choice[i] = prob[i, choice_rev1[i] ] #prob_choice as the probability of individual i chooses his/her actual choice
  }
  prob_choice[prob_choice >0.999999] = 0.999999
  prob_choice[prob_choice <0.000001] = 0.000001 # To prevent prob from going to negative or above one
  like = sum( log(prob_choice) )
  return(- like) #remember I already has a minus here
}


(246 -1)*2 # = 490 parameters to estimate
```

```
## [1] 490
```

```
# since it takes forever to optimize once, I choose to store the result of my first a
ttempt
#searchv = runif(490, -1, 1)

#result  = optim(searchv, fn = like_fun1, method = "BFGS",
#                  control = list(trace = 6, maxit = 3000),
#                  try5 = try5) #leave out "par" because "par" is what we want to est
imate
#first_estimate = result$par
#first_like = result$value #274486.6

#result  = optim(searchv, fn = like_fun1, method = "BFGS",
#                  control = list(trace = 6, maxit = 3000),
#                  try5 = try5, hessian = TRUE)
#second_estimate = result$par
#second_like = result$value

#final  value 261845.372652
#the second attempts takes more than two hours so I stop
```

```
#Because simply choosing random search value takes too long, I choose to use multinom
to guide me through searching value
options(scipen=999) #prevent scientific notations
library(nnet)

pack_res = multinom(choice_rev1 ~ score, data= try5)
```

```
## # weights:  738 (490 variable)
## initial  value 110106.630719
## iter  10 value 77126.386421
## iter  20 value 76742.396429
## iter  30 value 76741.591899
## iter  40 value 76740.563533
## iter  50 value 76739.306929
## iter  60 value 75832.858559
## iter  70 value 75565.009357
## iter  80 value 75551.569011
## iter  90 value 75551.026297
## iter 100 value 75479.863279
## final  value 75479.863279
## stopped after 100 iterations
```

```
pack_coef <- as.data.frame( coef(pack_res) )

pack_search <- c(pack_coef$`(Intercept)`, pack_coef$score)
# this vector list the 245 intercept estimates and then 245 choice:score estimates
```

```
result3  = optim(pack_search, fn = like_fun1, method = "BFGS",
                  control = list(trace = 6, maxit = 3000),
                  try5 = try5)
```

```
## initial  value 75479.863279
## iter  10 value 74080.776830
## iter  20 value 73937.258843
## iter  30 value 73662.665029
## iter  40 value 73621.332644
## iter  40 value 73621.332644
## final  value 73620.091048
## converged
```

```
result3$value #likelihood: - 73620.09
```

```
## [1] 73620.09
```

```
multi_param <- result3$par
multi_param
```

```
##   [1]   0.115981599189  -0.005916811908   0.231439077260   1.201919185951
##   [5]   1.335697369998  -0.000084289633  -9.254531603743   0.538253038017
##   [9]   0.437817946304   0.001948463244   0.197607799146   0.022075244622
##  [13]   0.006088014442  -0.026818875257   0.009079079109   0.034415830897
##  [17]   0.062937375801  -0.012190853407   0.031379769284   0.170816542943
##  [21]   0.194480345416  -0.040580996754   0.133089110520   1.397687752542
##  [25]   0.311827321251   0.514079912759  -0.095740396399  -0.012926611464
##  [29]   0.007546723927  -0.021501526967   0.016989986404   1.158467586359
##  [33]  -0.419833986207   0.000272880226  -1.176287369220   0.999059116725
##  [37]   0.281342177586   0.089435051745   0.132958221931   0.025885414972
##  [41]   0.016388300648  -0.011475801899   0.005367667969   0.133238729704
##  [45]   0.028836522940   0.048513072104  -0.007994191511  -0.003052856591
##  [49]   0.019264828384   0.051452486695   0.030542229889   0.014857141679
##  [53]   0.000706909087  -0.003052856591  -0.347150504330  -0.341144263119
##  [57]  -0.107938360098  -3.295486999427  -0.771100571336  -0.037682557248
##  [61]  -1.393158427552  -0.000819020253   0.002254321694   0.541065302438
##  [65]   0.335027860997   0.068254262217   0.099395233586   0.070318600184
##  [69]  -0.005212394013   0.016388300647  -3.603403632300  -3.116560635860
##  [73]  -0.002686502886 -16.380448436247  -0.015167996583   0.580429363528
##  [77]   0.299736726504   0.063494675062  -0.034266206030   0.241649590713
##  [81]   0.367253520486  -0.003347900860   0.133898006786   0.218569916685
##  [85]   0.089435051745   0.064767826949   0.182477016054   0.202690884019
##  [89]  -0.020049444580   0.162620046233  -0.023111230653  -0.026450331028
##  [93]   0.047053913863  -0.049809063566   0.160443904148   0.164944452290
##  [97]  -0.003052856591   0.144488588698   0.065928060118   0.020041702942
## [101]   0.009004375489   0.063492782455  -0.015167996582  -0.006969346948
## [105]  -0.007994191511  -0.007994191511   0.008273089686   1.150766309043
## [109]   0.827582654379   0.260422990284   0.644824905925   0.008999462236
## [113]   0.005359890218  -0.013680817990  -0.040874633068   0.007545406043
## [117]   0.024254724231  -0.013680817990  -0.010766259707  -0.008682157887
## [121]  -0.001571623341  -0.027171253295  -0.010066258577   0.034149594466
## [125]  -0.004075433299  -0.024617352047   0.035879378183   0.085897107311
## [129]   0.046617958123   9.128594298502   4.389954673792   0.671655751460
## [133]  -2.183285537762   0.775622778471   0.514257094745   0.139032641046
## [137]  -0.145900613414   0.060699334425   0.065893404611   0.014926700988
## [141]   0.051103002122   1.004832951673   0.601425703270   0.056965724606
## [145]   0.243878001481   0.085294025426   0.074204599731   0.025928660519
## [149]   0.014124017631   0.064041937893   0.006816516984   0.318269124545
## [153]   0.118754011282   0.024424678351   0.153032872139   0.064806405764
## [157]   0.053339524429   0.123494956361   0.081818674111  -0.007305845288
## [161]  -0.005916811909   0.014857141680   0.019311090841  -0.003052856591
## [165]  -0.003052856591   0.023368570903  -0.032238939469   0.006820364989
## [169]   0.014124017631   0.022075244623  -0.005212394013   1.276629250459
## [173]   0.394502880175   0.429922246702   0.292733542405  -0.013680817991
## [177]   0.024981228085   0.040097322844  -0.008682157887  -0.004500615793
## [181]  -0.001571623342   0.098683494476  -0.004500615793  -0.003780897391
## [185]   0.022517200442   0.017716434643   0.005367667969  -0.001571623341
## [189]  -0.002618453770  -0.015912957313   0.001478418366  -0.008415073230
## [193]   0.002254321694   0.092296810211   0.112789973798   0.037911729630
## [197]   0.869692697413   0.284418431869   0.050446451591   0.121701582324
## [201]   0.002462359151  -0.003052856591  -0.010066258577   0.014124017631
## [205]  -0.002316377561   0.121855963102   0.095779308634   0.020622285407
## [209]   0.221407158001   0.261186757376   0.225107488256   0.016388300648
## [213]   0.324294037694   0.000998604151   0.002374849604   0.014927484563
## [217]  -0.006614385555   0.027025268664  -0.010766259707   0.010464984281
## [221] -0.007692303765  0.008274689594 -0.021403860116 -0.000819020253
## [225]  0.001478418366 -0.003052856591  0.008156665943  0.010464984280
## [229]  0.379069606132  0.103841865007  0.006306867151  0.352480209955
## [233]  0.002254321695 -0.012190853408 -0.023679578517  0.071153512233
## [237]  0.023396604726  0.035025818722  0.007546723928  0.042112685818
## [241]  0.015026089954  0.125839322560  0.108073203279 -0.009157653007
## [245] -0.032939378867 -0.003058750926 -0.007791370275 -0.000751077652
## [249]  0.006144764359  0.003839123253  0.002318859120  0.032913996711
## [253]  0.001336246703  0.001194720909 -0.003861268627  0.000758939963
## [257] -0.005702510989 -0.006712328594 -0.007519128685 -0.005694855533
## [261] -0.002946020194 -0.003636658872 -0.007578978192 -0.004237831576
## [265] -0.001919734133 -0.002008767034 -0.007001003630 -0.003495215925
## [269]  0.002467867808  0.004343208987  0.000472975950  0.005264688141
## [273] -0.008164052097 -0.005652719309 -0.005527185554 -0.005685642948
## [277]  0.003907118731  0.006655572215 -0.006666998310  0.007348678558
## [281]  0.000901976373  0.000715207535 -0.002243734456 -0.000365244088
## [285] -0.005580898237 -0.005499014949 -0.009361450556 -0.005644818324
## [289] -0.002800219349 -0.003912391226 -0.005574127542 -0.007237760063
## [293] -0.007765919544 -0.006544300377 -0.005489260217 -0.004936222672
## [297] -0.006610273862 -0.007658183077 -0.007765919544  0.004827163665
## [301]  0.004463137008  0.000718415731  0.014520111761  0.007042266645
## [305]  0.001781047360  0.007667437398 -0.007723362182 -0.007665323400
## [309]  0.000255668130 -0.000366202083 -0.002357523214 -0.000909713820
## [313] -0.002887655421 -0.007762471157 -0.005499014949  0.018864211405
## [317]  0.016382481819  0.002467603972  0.051210569501 -0.007519454486
## [321]  0.001840468590  0.000516091397 -0.003276138668  0.000878907799
## [325] -0.000113140793 -0.000073223276 -0.005611721588 -0.001980183147
## [329] -0.002626763290 -0.002243734456 -0.004349981945 -0.002090026305
## [333] -0.001198456460 -0.005535241267 -0.001916519443 -0.007507807290
## [337] -0.006396602987 -0.005576736800 -0.005289526993 -0.002013207467
## [341] -0.001356093473 -0.007765919544 -0.002023231161 -0.004015129699
## [345] -0.005507104178 -0.006723756109 -0.003467246007 -0.007519454493
## [349] -0.006594361583 -0.007237760063 -0.007237760064 -0.005655326035
## [353]  0.001135455812  0.001180297897  0.001110827260  0.002762775797
## [357] -0.005657919834 -0.006707293061 -0.011767582808 -0.006312403459
## [361] -0.006720501647 -0.005709546126 -0.011767582813 -0.008568672519
## [365] -0.007350115628 -0.007749140095 -0.006394912793 -0.008141346552
## [369] -0.005034969090 -0.006626642208 -0.007082190369 -0.005745114800
## [373] -0.003237100999 -0.004076968364 -0.014149542722 -0.002887475234
## [377]  0.000041908958  0.015463570870  0.003393434065  0.002446692686
## [381] -0.002816563566  0.003425865461 -0.000331441418 -0.002729225795
## [385] -0.005514552684 -0.003031381739  0.000679698422 -0.000004170458
## [389] -0.003446184912 -0.000584324392 -0.003710782571 -0.004382338682
## [393] -0.003904005448 -0.006636826772 -0.004347506352 -0.006716826650
## [397] -0.001751987371 -0.002759121333 -0.005562085811 -0.002957880324
## [401] -0.002686677282 -0.003434448510 -0.003992316423 -0.002648372346
## [405] -0.007366447826 -0.007791370275 -0.006610273861 -0.005501232315
## [409] -0.007765919544 -0.007765919544 -0.002592933107 -0.003156299347
## [413] -0.005650099366 -0.006636826772 -0.005702510989 -0.007762471157
## [417]  0.000388594945  0.000360743252 -0.001150787108  0.003557909042
## [421] -0.011767582820 -0.005711864627 -0.004057905212 -0.007350115625
## [425] -0.007753785230 -0.007749140094 -0.004024854864 -0.007753785230
## [429] -0.007760141723 -0.003571420215 -0.005688090459 -0.005644818325
## [433] -0.007749140095 -0.005788999199 -0.007615228473 -0.007646039601
## [437] -0.006576533832 -0.007665323400 -0.002771915368 -0.003422961194
## [441] -0.004259528261  0.001195590722  0.002150570077 -0.003245158325
## [445]   0.002388472029  -0.005634077899  -0.007765919544  -0.008141346555
## [449]  -0.006636826772  -0.007763175040  -0.002531668580  -0.000908955960
## [453]  -0.005697755338  -0.000570292215  -0.001306558276  -0.001691635060
## [457]  -0.005499014949   0.000502653352  -0.006673057207  -0.004729834856
## [461]  -0.003177500962  -0.007720302311  -0.004223419001  -0.008568672518
## [465]  -0.006717905204  -0.006585576027  -0.006722972192  -0.006420827721
## [469]  -0.007723362182  -0.007646039601  -0.007765919544  -0.004781918783
## [473]  -0.006717905204  -0.000402799923  -0.002075722084  -0.003868934631
## [477]   0.000760215929  -0.007665323400  -0.007578978196  -0.005514508176
## [481]  -0.002366234462  -0.004211436467  -0.004044154660  -0.005652719309
## [485]  -0.002647146861  -0.003882653064  -0.003468829325  -0.000910018532
## [489]  -0.005587727031  -0.006382495254
```

```
#Marginal effect

# theory: p_ij(Beta_j - sum (p_il*Beta_l) )
# use the truncated likelihood function to compute probability

out_fun <- function(par, try5) {
  choice_rev1 = try5$choice_rev1
  score = try5$score

  n_i = nrow(try5) #should be 20,000 students
  n_j = length(unique(try5$choice_rev1)) #246 choices
  out = mat.or.vec( n_i,n_j )

  n_jref = n_j - 1

  out[,1] = 0

  par_set1 = par[1:n_jref]
  par_set2 = par[ (n_jref+1) : (2*n_jref) ]

  for (j in 2:n_j) {
    out[,j] = par_set1[(j-1)] + par_set2[j-1] * score
  }
  return(out)
}

out <- out_fun(multi_param, try5)
prob = exp(out)
prob = sweep(prob, MARGIN=1, FUN="/", STATS=rowSums(prob))
prob = as.data.frame.matrix(prob)

for (i in 1:20000) {prob$prob_choice[i] = prob[i, try5$choice_rev1[i] ]}

for (h in 1:20000)  {prob$beta_j[h] = multi_param[ (try5$choice_rev1[h]+ 245) ]}
```

```
multprob <- prob[,-1] #since the first choice should be the reference group, now we have 245 columns instead of 246

score_param <- multi_param[246:490]
score_param <- as.matrix(score_param)  #to make its dimension: (j-1)*1

multprob <- multprob %>%
  mutate(B_i_bar = 0)

for (i in 1: length(multprob) ) {
  multprob$B_i_bar[i] <- sum( as.matrix( multprob[i,1:245] ) %*% score_param)
}

multprob <- multprob %>%
  mutate(marginal = prob_choice * (beta_j - B_i_bar) )

multprob$marginal #This is the marginal effect
```

I didn’t show all the marginal effects here since I have 20,000 of them

```
##   [1] -0.02794337775747 -0.00031348761635 -0.00031760636927 -0.02684419221827
##   [5] -0.00032158224861 -0.00032158224861 -0.02573717914477 -0.02573717914477
##   [9] -0.02573717914477 -0.02518173006647 -0.02518173006647 -0.00174034877542
##  [13] -0.02518173006647 -0.02518173006647 -0.02462553662978 -0.02406902330754
##  [17] -0.00508867075196 -0.02406902330754 -0.00505231520767 -0.02351262240702
##  [21] -0.00033895788308 -0.02351262240702 -0.00124314518203 -0.02295677286352
##  [25] -0.00501318852165 -0.00034186180232 -0.02295677286352 -0.00112699407181
##  [29] -0.02240191896467 -0.00123477164052 -0.00492674613926 -0.00034702171652
##  [33] -0.02184850901176 -0.02184850901176 -0.02184850901176 -0.02184850901176
##  [37] -0.02184850901176 -0.00013439444705 -0.00076262411425 -0.00076262411425
##  [41] -0.00034925990741 -0.00122873506734 -0.02129699392489 -0.00114247579636
##  [45] -0.02074782580013 -0.00186532576269 -0.00186532576269 -0.00482966260853
##  [49] -0.00035125905225 -0.00035125905225 -0.02020145642700  0.00002629032306
##  [53] -0.00001192135328 -0.00477726073107 -0.00035301132019 -0.00035301132019
##  [57] -0.02020145642700 -0.00115517103352 -0.00115517103352 -0.02020145642700
##  [61] -0.01965833577569 -0.01965833577569 -0.00472236876281 -0.01965833577569
##  [65] -0.00472236876281 -0.01965833577569 -0.01965833577569 -0.00116042181611
##  [69] -0.01965833577569 -0.00120304305037 -0.00120304305037 -0.00466505761856
##  [73] -0.01911891046382 -0.01911891046382 -0.00006643722259 -0.00189389888553
##  [77] -0.00466505761856 -0.00189389888553 -0.01911891046382 -0.01911891046382
##  [81] -0.00119191991931 -0.01911891046382 -0.01911891046382 -0.00460540474854
##  [85] -0.01858362221312 -0.00190091374173 -0.00037919332884 -0.00460540474854
##  [89] -0.00035671647606 -0.01858362221312 -0.01858362221312 -0.01858362221312
##  [93] -0.01858362221312 -0.00014602883783 -0.00454349392013 -0.01805290630673
##  [97] -0.00190662695880 -0.00454349392013 -0.00117157635708 -0.00083815302092
## [101] -0.00017155159279 -0.00191101506862 -0.00191101506862 -0.00191101506862
## [105] -0.00022072471842 -0.00191101506862 -0.00191101506862 -0.00447941496051
## [109] -0.00035783356501 -0.00035783356501 -0.00017584255744 -0.01752719005817
## [113] -0.00117371399421 -0.01752719005817 -0.01752719005817 -0.01752719005817
## [117] -0.00026641632248 -0.00441326346134 -0.00084715067049 -0.01700689130313
## [121] -0.00113459186584 -0.01700689130313 -0.00441326346134 -0.00441326346134
## [125] -0.00441326346134 -0.00441326346134 -0.00441326346134 -0.00035797165565
## [129] -0.00009480326656 -0.00117504227271 -0.00117504227271 -0.01700689130313
## [133] -0.01700689130313 -0.01700689130313 -0.01649241692517 -0.01649241692517
## [137] -0.00117555283733 -0.00117555283733 -0.00111708555623 -0.01649241692517
## [141] -0.01649241692517 -0.00191573885678 -0.00434514044704 -0.00033344970114
## [145] -0.00434514044704 -0.00001225841688 -0.00111708555623 -0.01649241692517
## [149] -0.01649241692517 -0.01649241692517 -0.00111708555623 -0.01649241692517
## [153] -0.01649241692517 -0.01649241692517 -0.00014813131431 -0.00191604474769
## [157] -0.00427515200851 -0.00427515200851 -0.00427515200851 -0.00427515200851
## [161] -0.00427515200851 -0.00427515200851 -0.00037238142139 -0.00427515200851
## [165] -0.00427515200851 -0.00427515200851 -0.00035738983274 -0.00035738983274
## [169] -0.01598416142652 -0.01598416142652 -0.01598416142652 -0.01598416142652
## [173] -0.01598416142652 -0.00109833681510 -0.00109833681510 -0.01598416142652
## [177] -0.01598416142652 -0.00030294509376 -0.00030294509376 -0.01548250555470
## [181] -0.01548250555470 -0.00420340890482 -0.00191496617041 -0.00191496617041
## [185] -0.00420340890482 -0.00420340890482 -0.00019492899602 -0.00018492367654
## [189] -0.01548250555470 -0.00107836604518 -0.01548250555470 -0.00413002613535
## [193] -0.00087771489385 -0.00087771489385 -0.00087771489385 -0.00413002613535
## [197] -0.00413002613535 -0.00087771489385 -0.00087771489385 -0.00117212732139
## [201] -0.00117212732139 -0.01498781499529 -0.01498781499529 -0.00191249744489
## [205] -0.00191249744489 -0.00191249744489 -0.00043865316930 -0.00191249744489
## [209] -0.00043865316930 -0.00413002613535 -0.00038089137451 -0.00038089137451
## [213] -0.00043865316930 -0.00105719747479 -0.00117212732139 -0.01498781499529
## [217] -0.01498781499529 -0.01498781499529 -0.00105719747479 -0.00105719747479
##  [221] -0.01498781499529 -0.01498781499529 -0.00031001064393 -0.00015609635687
##  [225] -0.00405512248544 -0.01450043914094 -0.00103485909943 -0.00103485909943
##  [229] -0.01450043914094 -0.01450043914094 -0.00018479255068 -0.00007781114493
##  [233] -0.00044668726140 -0.00020591213339 -0.00190863669735 -0.00190863669735
##  [237] -0.00190863669735 -0.00405512248544 -0.00405512248544 -0.00405512248544
##  [241]  0.00000723329205 -0.00405512248544 -0.00103485909943 -0.01450043914094
##  [245] -0.01450043914094 -0.01450043914094 -0.00103485909943 -0.01450043914094
##  [249] -0.00297368189281  0.00132814293567 -0.00297368189281 -0.00297368189281
##  [253]  0.00013717426189  0.00013717426189  0.00019406476426  0.00019406476426
##  [257]  0.00134159285738  0.00134159285738 -0.00290818871153  0.00134159285738
##  [261] -0.00290818871153  0.00025490146248  0.00025490146248  0.00000312400692
##  [265]  0.00025490146248  0.00025490146248 -0.00003561217792  0.00019406476426
##  [269]  0.00019406476426  0.00019406476426  0.00013717426189  0.00019406476426
##  [273]  0.00003410563300  0.00010951754033 -0.00290818871153  0.00006491161555
##  [277]  0.00000260459411  0.00011452703168 -0.00290818871153 -0.00290818871153
##  [281] -0.00290818871153  0.00134159285738 -0.00290818871153 -0.00290818871153
##  [285] -0.00290818871153  0.00010951754033 -0.00290818871153 -0.00290818871153
##  [289] -0.00290818871153  0.00008660448288  0.00000520252811  0.00019321484223
##  [293]  0.00019321484223  0.00019321484223  0.00019321484223 -0.00284295692395
##  [297] -0.00000011713500 -0.00284295692395  0.00135461623810  0.00135461623810
##  [301]  0.00025825262614  0.00025825262614  0.00025825262614 -0.00005522829825
##  [305] -0.00003716479705  0.00019321484223  0.00019321484223  0.00019321484223
##  [309]  0.00019321484223  0.00011085544322  0.00014027883729  0.00011614223071
##  [313]  0.00000266802504  0.00001345385880 -0.00004128840750  0.00135461623810
##  [317]  0.00011085544322  0.00135461623810  0.00135461623810  0.00135461623810
##  [321]  0.00135461623810 -0.00284295692395 -0.00284295692395  0.00001730059523
##  [325]  0.00000531165914  0.00019228852583 -0.00277803084877  0.00011216295788
##  [329]  0.00136719640205  0.00000810333365  0.00136719640205  0.00011330282660
##  [333]  0.00026153887749 -0.00003876895423  0.00026153887749  0.00026153887749
##  [337]  0.00000070674437  0.00026153887749  0.00026153887749  0.00019228852583
##  [341]  0.00014339393168  0.00014339393168  0.00011773115676 -0.00004214106035
##  [345] -0.00277803084877 -0.00277803084877 -0.00277803084877 -0.00277803084877
##  [349]  0.00136719640205  0.00136719640205 -0.00277803084877  0.00136719640205
##  [353] -0.00277803084877  0.00000542081555  0.00003668136790  0.00019128676654
##  [357]  0.00019128676654  0.00014651701382  0.00019128676654  0.00019128676654
##  [361]  0.00137931737546 -0.00271345434968  0.00011343852096 -0.00271345434968
##  [365]  0.00137931737546 -0.00271345434968 -0.00271345434968  0.00011343852096
##  [369] -0.00271345434968 -0.00271345434968 -0.00271345434968 -0.00271345434968
##  [373]  0.00026475638048  0.00026475638048  0.00026475638048  0.00026475638048
##  [377]  0.00026475638048 -0.00005778591403 -0.00005778591403  0.00026475638048
##  [381]  0.00019128676654  0.00019128676654  0.00007013302210  0.00019128676654
##  [385]  0.00014651701382  0.00019128676654 -0.00271345434968  0.00011343852096
##  [389]  0.00011343852096  0.00026475638048  0.00011929200278  0.00011929200278
##  [393] -0.00004299336686 -0.00271345434968  0.00011343852096 -0.00271345434968
##  [397]  0.00001816965669  0.00019021065679  0.00019021065679  0.00019021065679
##  [401]  0.00019021065679  0.00019021065679  0.00139096393745  0.00139096393745
##  [405] -0.00264927071048 -0.00264927071048  0.00026790139680  0.00026790139680
##  [409]  0.00026790139680  0.00026790139680  0.00026790139680  0.00019021065679
##  [413]  0.00019021065679  0.00014964551587  0.00012082299914  0.00002072582977
##  [417] -0.00004384457020 -0.00004384457020 -0.00004384457020 -0.00264927071048
##  [421]  0.00139096393745 -0.00264927071048 -0.00264927071048 -0.00264927071048
##  [425] -0.00264927071048 -0.00264927071048  0.00011468061685 -0.00006555468463
##  [429]  0.00003842533320  0.00018906142687  0.00015277684027 -0.00258552251235
##  [433] -0.00000083841110  0.00011588778231  0.00140212166790  0.00140212166790
##  [437] -0.00006036092363  0.00015277684027 -0.00258552251235  0.00027097029721
##  [441]  0.00027097029721  0.00027097029721  0.00000351082278  0.00027097029721
## [19933] -0.00011194833589  0.00019931487566 -0.00000020953994  0.00003669885369
## [19937] -0.00000159918108  0.00008725894682 -0.00002633310302 -0.00000226921252
## [19941]  0.00000282861558 -0.00011194833589  0.00000337947279  0.00000626519532
## [19945] -0.00000161736761 -0.00000679495258  0.00009908089747  0.00009908089747
## [19949]  0.00026562802288  0.00009908089747 -0.00000679495258 -0.00000349113799
## [19953] -0.00010673399336  0.00017601180430  0.00019923629500  0.00008089918875
## [19957]  0.00003563557050  0.00026562802288  0.00017601180430  0.00008089918875
## [19961]  0.00084822536765  0.00084822536765  0.00000628157393  0.00008614314622
## [19965]  0.00000628157393 -0.00010924643979 -0.00000300143679 -0.00000208343880
## [19969]  0.00021738433015  0.00000284677752 -0.00041134467841  0.00000284677752
## [19973]  0.00021738433015  0.00000284677752 -0.00001474280774 -0.00041134467841
## [19977] -0.00000226940978 -0.00000003404537 -0.00000003404537 -0.00041134467841
## [19981]  0.00008089918875 -0.00000003404537  0.00019923629500 -0.00000418911833
## [19985]  0.00026562802288  0.00008614314622  0.00000284677752 -0.00006750549376
## [19989] -0.00000021030110  0.00026562802288 -0.00000246588787  0.00002444805473
## [19993]  0.00021738433015  0.00000284677752 -0.00000098137301 -0.00001066230480
## [19997] -0.00000098137101 -0.00000257990599  0.00084822536765  0.00084822536765
```

#Exercise 6 Conditional Logit

dependent variable: first choice independent variable: school quality *Use conditional logit

```
#In conditional logit, the beta estimate does not vary by choice. Hence, I only need to estimate two coefficients: intercept and school quality

colnames(name_list) <- c("first_choice_name", "choice_rev1")
colnames(new_cutqua) <- c("first_choice_name", "cutoff", "quality")
name_list <- name_list %>%
  left_join(new_cutqua, by = "first_choice_name")

try5 <- cbind(try5, name_list$quality)
try5 <- rename(try5, quality = V2) #finally put school quality in the dataset

cond_choice_rev1 <- try5$choice_rev1
cond_quality     <- try5$quality

Con_fun1 <- function(par, cond_choice_rev1, cond_quality) {
  choice_rev1 = cond_choice_rev1
  quality = cond_quality

  n_i = nrow(data)
  n_j = length(unique( choice_rev1 )) #246 choices
  out = mat.or.vec( n_i,n_j )
  #This out should eventually contain the imagined utility for every individual and their potential choice

  #remember to omit a choice as the reference choice
  n_jref = n_j - 1

  #what is the restriction for conditional logit?
  out[,1] = 0

  #parameter set for every right-hand side variables and intercept

  intercept = par[1:n_jref]  #intercept
  par_qua = par[ (n_jref+1) ] #the score coefficient. In conditional logit, the Beta does not vary by choice

  for (i in 1:n_i) {
    out[i,] = par_qua * quality[i] #first deal with quality effect
  }

  for (j in 2:n_j) {
    out[,j] = out[,j] + intercept[ (j-1) ] #then I add corresponding intercept to each column
  }

  prob = exp(out)
  prob = sweep(prob, MARGIN=1, FUN="/", STATS=rowSums(prob))

  prob_choice = NULL
for (i in 1:n_i){
    prob_choice[i] = prob[i, choice_rev1[i] ] #prob_choice as the probability of individual i chooses his/her actual choice
}

  prob_choice[prob_choice >0.999999] = 0.999999
  prob_choice[prob_choice <0.000001] = 0.000001

  like = sum( log(prob_choice) , na.rm = T) #When I test it I found out two numbers a
re NA so initially I cannot sum
  return(- like) #remember I already has a minus here
}
```

```
#find ideal searching values by using package

library(mlogit)
```

```
## Loading required package: dfidx
```

```
##
## Attaching package: 'dfidx'
```

```
## The following object is masked from 'package:stats':
##
##     filter
```

```
library(tidyr)

test_dat <- try5 %>%
  mutate(first_choice = choice_rev1) %>% #keep a copy of choice variable
  pivot_wider(names_from = choice_rev1, values_from = quality, values_fill = 0)

#transform my data to have every choice as a column

for (i in 37: 282) {
  test_dat[,i] = max(test_dat[,i])
}

for (r in 37:282){
  colnames(test_dat)[r]= paste0("quality_", colnames( test_dat[,r] ))
}

mloDat = mlogit.data(test_dat, varying = 37:282, shape = "wide", sep = '_',
                               choice = "first_choice")

#pack_cond <- mlogit(first_choice ~  quality , data = mloDat)

#Since my computer cannot handle this operation, below are codes that I think should
 work but I cannot run them without the mlogit result

#pack_condcf <- as.data.frame( coef(pack_cond) )
#cond_search <- c(pack_coef$`(Intercept)`, pack_coef$quality)
```

```
#This takes forever to run.

#cond_result  = optim(cond_search, fn = Con_fun1, method = "BFGS",
#                   control = list(trace = 6, maxit = 3000),
#                   try5 = try5)
#cond_par <- cond_result$par
```

Here I use a subsample to complete optimization

```
# samp_try5 <- try5[ sample( nrow(try5), 100) , ] #sample 100 rows at random

# searchv <- runif(length(unique( samp_try5$choice_rev1) ), -1, 1) #num of unique cho
ice -1 + 1 (quality coefficient)
# samp_result  = optim(searchv, fn = Con_fun1, method = "BFGS",
#                   control = list(trace = 6, maxit = 3000),
#                   cond_choice_rev1 = cond_choice_rev1, cond_quality = cond_quality,
#                   )
# My attempt fails because of this error: Error in matrix(0, nr, nc) : non-numeric ma
trix extent
```

Conditional logit marginal effect Since I cannot produce the result in previous operation, here is my plan to produce marginal effect.

```
#Marginal effect

# theory: p_ij(delta_ijk - p_ik)* Beta
Conmar_fun <- function(par, try5) {
  choice_rev1 = try5$choice_rev1
  quality = try5$quality

  n_i = nrow(try5)
  n_j = length(unique( try5$choice_rev1) ) #246 choices
  out = mat.or.vec( n_i,n_j )

  n_jref = n_j - 1

  out[,1] = 0

  intercept = par[1:n_jref]  #intercept
  par_qua = par[ (n_jref+1) ] #the score coefficient. In conditional logit, the Beta does not vary by choice

  for (i in 1:n_i) {
    out[i,] = par_qua * quality[i] #first deal with quality effect
  }

  for (j in 2:n_j) {
    out[,j] = out[,j] + intercept[ (j-1) ] #then I add corresponding intercept to each column
  }
  prob = exp(out)
  prob = sweep(prob, MARGIN=1, FUN="/", STATS=rowSums(prob))
  prob = as.data.frame.matrix(prob)

  for (i in 1:nrow(try5))  {prob$prob_choice[i] = prob[i, try5$choice_rev1[i] ]} #This is my P_ij
  #now I have to compute (delta_ijk - pik), this is a vector
  #pik represents the probability in one row
  #delta_ijk = 1 if that is prob_choice, if alternative, than 0

  pik = prob[,- ncol(prob)] #because the last column is the prob_choice I just created
  delta_ijk = pik #just copy this prob matrix (dimension: n_i*n_j)

  for (i in 1 : nrow(try5)) {
  delta_ijk[i,] = ifelse(delta_ijk[i,] == prob$prob_choice[i], 1, 0) #If the probability matches choice probability, I consider that as j = k, so 1. If probability does not match, it means j != k, so 0.
  }
  second_term = delta_ijk - pik #matrix subtraction

  marginal = mat.or.vec(n_i, n_j)

  for (i in 1: nrow(try5)) {
  marginal[i, ] = prob$prob_choice[i]* second_term[i,] * par[length(par)]
  #Why par[length(par)]?The last parameter should be the "quality" coefficient
  }
  return(marginal)
}

#Conmar_fun(cond_par, try5)
```

Exercise 7 Counterfactual simulation

excluding choices where the program is “Others”

```
#Q1

# I think we should use the second model, which is the conditional logit. To explain
 this I will give an example: For those students choosing majors that "yield better f
uture income" (a program characteristic), if they are told they can no longer choose
 to study "engineering" in college, they will change their preference to some other p
rogram that give them similar income.

#What I am saying is that limiting options affect and limit the choice characteristic
s. Thus, studying choice exclusion should use conditional logit, which deals with the
effect choice characteristics.

# Q2
# Excluding choices with "others" mean that program called "others" should yield no u
tility for individuals. I can do this by setting those variable utility columns (in t
he utility matrix in likelihood function) to 0

#First, recall I have made a school-program factor number list. I will use the transf
ormed version later in my function.
library(stringr)

others_pgm <- name_list %>%
  filter( str_detect(first_choice_name, "others") == T )

others_pgm <- others_pgm[!duplicated( others_pgm$first_choice_name), ] #remove duplic
ate
others_pgm <- select(others_pgm, first_choice_name, choice_rev1)
others_fac_num <- others_pgm$choice_rev1 #This is the vector of the factor number of
 programs called "Others"
```

```
#Using part of my conditional logit function

Prob_mat <- function(par, cond_choice_rev1, cond_quality, others_fac_num) {
  choice_rev1 = cond_choice_rev1
  quality = cond_quality

  n_i = nrow(data)
  n_j = length(unique( choice_rev1 )) #246 choices
  out = mat.or.vec( n_i,n_j )

  n_jref = n_j - 1

  out[,1] = 0

  intercept = par[1:n_jref]
  par_qua = par[ (n_jref+1) ]

  for (i in 1:n_i) {
    out[i,] = par_qua * quality[i]
  }

  for (j in 2:n_j) {
    out[,j] = out[,j] + intercept[ (j-1) ]
  }

  prob = exp(out)

  #Since some of the choices are "Others", for this question, we should set these utilities to 0 here. Remember in conditional logit, we have j-1 intercepts as columns.
  #The first column should represent as.number(as.factor (choice)) = 1. That is, if I know "Others" corresponding factor number, I can locate those columns and restrict those column to 0

  for (u in others_fac_num) {   # I made this "others_fac_num in the previous chunk
    prob[,u] = 0.00000001       # Prevent dividing 0
  }

  prob = sweep(prob, MARGIN=1, FUN="/", STATS=rowSums(prob))
  return(prob)
}

#Since I don't have conditional logit's estimate, I will show how I am going to do this

#Prob_mat_exclude <- Prob_mat(par, cond_choice_rev1, cond_quality, others_fac_num)
```

In Q3, I will also show what I am going to do if I have the estimates

```r
# Q3

Q3_fun <- function(par, try5) {
  choice_rev1 = try5$choice_rev1
  quality = try5$quality

  n_i = nrow(try5)
  n_j = length(unique( try5$choice_rev1) ) #246 choices
  out = mat.or.vec( n_i,n_j )

  n_jref = n_j - 1

  out[,1] = 0

  intercept = par[1:n_jref]  #intercept
  par_qua = par[ (n_jref+1) ] #the score coefficient. In conditional logit, the Beta does not vary by choice

  for (i in 1:n_i) {
    out[i,] = par_qua * quality[i] #first deal with quality effect
  }

  for (j in 2:n_j) {
    out[,j] = out[,j] + intercept[ (j-1) ] #then I add corresponding intercept to each column
  }
  prob = exp(out)
  prob = sweep(prob, MARGIN=1, FUN="/", STATS=rowSums(prob))
  return(prob)
}

# Origin_prob <- Q3_fun(cond_par, try5)
# Prob_change <- Prob_mat_exclude - Origin_prob    #I think they should be of same dimension

# Prob_change
```