```
##### Assignment 2 Data #####
# 1/28/2020
# Bahar Zafer

### Exercise 1 Basic characteristics ###

library(readstata13)
library(xtable)
library(plyr)
library(dplyr)
library(ggplot2)
library(data.table)

dat = read.dta13("guiide_master_file_anonymized_full.dta")

# finding number of rows and columns
> nrow(dat)
[1] 42974
> ncol(dat)
[1] 3332

# sinking descriptive characteristics of variables into "summary.txt"
> sink("summary.txt")
> sapply(dat, FUN = summary)
> sink()

# finding an individual identifier
> varnames = names(dat)
> varnames[grep("id", varnames)]
[1] "anon_studentid"                "schoolid"
"stud_fu1_seen_video"
[4] "pg_bl_middle_income"           "schl_bl_confident"
"masterid_merge"
[7] "SHS_TotalCandidates"           "TotalCandidates_1"
"TotalCandidates_2"
[10] "TotalCandidates_3"             "TotalCandidates_4"
"a_when_decide"
[13] "a_see_vid"                     "a_idea"
"b_when_decide"
[16] "b_see_vid"                     "b_idea"
"alladmin_jssid"
[19] "alladmin_SHS_TotalCandidates" "alladmin_TotalCandidates_1"
"alladmin_TotalCandidates_2"
[22] "alladmin_TotalCandidates_3"   "alladmin_TotalCandidates_4"

# "anon_student_id" is likely to be the individual identifier

# checking if there are duplicates
> duplicates = which(duplicated(dat$anon_studentid) == 1)
> duplicates #returns index values
[1]  1001  1188  1397  2780  5127  5329  5333  5364  5965  7355  8258  8558 10226 10266
10493 10500 10759
[18] 10774 11516 12606 13169

# removing duplicates from data
dat = dat[-duplicates, ]
> which(duplicated(dat$anon_studentid) == 1)
integer(0)

### Exercise 2 Small Bases ###

> varnames[1]
[1] "anon_studentid"      # individual identifier, "anon_studentid" is the first
variable.

# dataset with ndividual identifier and all variables starting with "stud_base"
> dat.surv = dat[, c(1, grep("stud_base", varnames))]

# dataset with ndividual identifier and all variables starting with "alladmin"
```

```
> dat.admin = dat[, c(1, grep("alladmin", varnames))]

# dataset of other variables
> dat.rest = dat[, -c(grep("stud_base", varnames), grep("alladmin", varnames))]

### Exercise 3 Consistency ###

> gender_table = table(dat.surv$stud_base_GENDER, dat.admin$alladmin_GENDER)
> age_table = table(dat.surv$stud_base_age, dat.admin$alladmin_age)

# off-diagonal values in gender_table and age_table are inconsistent
> inconsistent_age_num = sum(age_table[upper.tri(age_table, diag = FALSE)]) +
sum(age_table[lower.tri(age_table, diag = FALSE)])
> inconsistent_gender_num = sum(gender_table[upper.tri(gender_table, diag = FALSE)]) +
sum(gender_table[lower.tri(gender_table, diag = FALSE)])
> inconsistent_gender_num
[1] 27091
> inconsistent_age_num
[1] 12780

> varnames[grep("bece",varnames)]
[1] "stud_base_taken_mock_bece"   "stud_base_bece_likely"
"stud_base_bece_best"
[4] "stud_base_bece_worst"        "stud_base_choice_1_bece"
"stud_base_choice_2_bece"
# only first 6 results are given.
# We will use "stud_base_bece_best", "stud_base_bece_likely" and "stud_base_bece_worst"
# checking if there is any inconsistency in the answers

inconsistent_cases = which(dat$stud_base_bece_worst > dat$stud_base_bece_best)
over_confidence_cases = which(dat$stud_base_bece_likely > dat$stud_base_bece_best)
under_confident_cases = which(dat$stud_base_bece_worst > dat$stud_base_bece_likely)

### Exercise 4 Balance ###

# finding the treatment variables
> varnames[grep("treat", varnames)]
[1] "treatgroup"           "stud_base_treatgroup" "alladmin_treatgroup"

# checking if gender is balanced across "treatgroups"
> test1_dat = table(dat$GENDER, dat$treatgroup)
> test1_dat
      1    2    3    4    5    6
      5  593    0  451    0  385
F     9 2135    3 1979    2 2184
M    15 2135    1 2219    0 2295

# testing whether female-male distribution is balanced across treatment 2 and 4
> test1_1 = prop.test(test1_dat[c(2, 3), c(2, 4)], p=c(0.5, 0.5))
> test1_1
2-sample test for given proportions with continuity correction

data:  test1_dat[c(2, 3), c(2, 4)], null probabilities c(0.5, 0.5)
X-squared = 7.422, df = 2, p-value = 0.02445
alternative hypothesis: two.sided
null values:
  prop 1 prop 2
0.5    0.5
sample estimates:
  prop 1    prop 2
0.5189596 0.4903537

# proportions are close to 0.5 but we can reject the null hypothesis at 0.05 percent
level.
# distribution is not balanced.

# same test across treatment 2, 4 and 6
> test1_2 = prop.test(test1_dat[c(2, 3), c(4, 6)], p=c(0.5, 0.5))    # p-value = 0.00362
> test1_3 = prop.test(test1_dat[c(2, 3), c(2, 6)], p=c(0.5, 0.5))    # p-value = 0.04415
```

```
125
126
127 # checking if age is balanced across "treatgroups"
128 > test2_dat = table(dat$stud_base_age, dat$treatgroup)
129
130 # testing if each group of age (from 14 to 18) is balanced across treatment 2, 4 and 6
131
132 > test2_1 = prop.test(test2_dat[14:18, c(2, 6)], p=c(0.5, 0.5, 0.5, 0.5, 0.5))
133 > test2_1
134 5-sample test for given proportions without continuity correction
135
136 data:  test2_dat[14:18, c(2, 6)], null probabilities c(0.5, 0.5, 0.5, 0.5, 0.5)
137 X-squared = 11.866, df = 5, p-value = 0.03667
138 alternative hypothesis: two.sided
139 null values:
140   prop 1 prop 2 prop 3 prop 4 prop 5
141 0.5    0.5    0.5    0.5    0.5
142 sample estimates:
143   prop 1    prop 2    prop 3    prop 4    prop 5
144 0.5027408 0.4855282 0.5485294 0.5607477 0.5666667
145
146 > test2_2 = prop.test(test2_dat[14:18, c(2, 4)], p=c(0.5, 0.5, 0.5, 0.5, 0.5))  #
    p-value = 0.001667
147 > test2_3 = prop.test(test2_dat[14:18, c(4, 6)], p=c(0.5, 0.5, 0.5, 0.5, 0.5))  #
    p-value = 0.5455
148
149 # checking if SHS region is balanced across "treatgroups"
150 > test3_dat = table(dat$SHSregioncode, dat$treatgroup)
151
152 # Balance tests across treatment 2, 4, and 6
153 > test3_1 = prop.test(test3_dat[,c(2, 4)])
154
155 > test3_1
156 12-sample test for equality of proportions without continuity correction
157
158 data:  test3_dat[, c(2, 4)]
159 X-squared = 87.801, df = 11, p-value = 4.49e-14
160 alternative hypothesis: two.sided
161 sample estimates:
162   prop 1    prop 2    prop 3    prop 4    prop 5    prop 6    prop 7    prop 8    prop
      9   prop 10
163 0.5680077 0.4842105 0.6190476 0.4767442 0.5581395 0.4761905 0.4943332 0.7151163
    0.9000000 0.7364341
164 prop 11   prop 12
165 0.4928910 0.6071429
166
167 > test3_2 = prop.test(test3_dat[,c(2, 6)])    # p-value < 2.2e-16
168 > test3_3 = prop.test(test3_dat[,c(4, 6)])    # p-value = 4.178e-08
169
170 ### Exercise 5 Recoding and Histogram ###
171
172 # searching for the variable of the highest education level desired
173 > varnames[grep("educ",varnames)]
174
175 # Recoding the variable, "stud_educ_want", using its levels.
176 > Educ_want = dat$stud_base_educ_want
177 > Educ_want = as.factor(Educ_want)
178 > levels(Educ_want) = c("Junior high school", "Technical or vocational training",
    "Senior high school", "nursing or teacher training", "Polytechnic", "University")
179
180 # the histogram of desired education by gender
181 > par(mfrow = c(1, 2))
182 > hist(dat$stud_base_educ_want[dat$stud_base_GENDER == "F"], xlab = "Female", main =
    "Desired Education")
183 > hist(dat$stud_base_educ_want[dat$stud_base_GENDER == "M"], xlab = "Male", main =
    "Desired Education")
184
185 ### Exercise 6 Manipulating the Data ###
186
```

```
library(reshape2)

# finding variables related with school/program choice
varnames[grep("mychoice",varnames)]
# recovered variables of choice:
# stud_base_mychoice_1_pgm, stud_base_mychoice_2_pgm, stud_base_mychoice_3_pgm,
stud_base_mychoice_4_pgm

# creating a dataset of program choices
> my_dat1 = cbind(dat$anon_studentid, dat$stud_base_mychoice_1_pgm,
dat$stud_base_mychoice_2_pgm, dat$stud_base_mychoice_3_pgm, dat$stud_base_mychoice_4_pgm)
> colnames(my_dat1) = c("id", "choice1", "choice2", "choice3", "choice4")
> my_dat1 = as.data.frame(my_dat1)
> my_dat1 = melt(my_dat1, id = c("id"))

# frequency table of programs by ranked choice number
> choiceFreq_programs = table(my_dat1$variable, my_dat1$value)
> choiceFreq_programs
              AGRICULTURAL SCIENCE BUSINESS
choice1 34465                  105      696
choice2 35018                  125      739
choice3 35407                  120      714
choice4 35834                  131      602

        GENERAL ARTS HOME ECONOMICS OTHER SCIENCE
choice1         3018           1565   510    1640
choice2         2802           1392   454    1404
choice3         2731           1282   458    1252
choice4         2614           1239   426    1146

        TECHNICAL SKILLS VISUAL ARTS
choice1              272         682
choice2              251         768
choice3              248         741
choice4              263         698

# creating a dataset of region choices
> varnames[grep("region",varnames)]

> my_dat2 = cbind(dat$anon_studentid, dat$stud_fu1_mychoice1_region,
dat$stud_fu1_mychoice2_region, dat$stud_fu1_mychoice3_region,
dat$stud_fu1_mychoice4_region)
> colnames(my_dat2) = c("id", "regchoice1", "regchoice2", "regchoice3", "regchoice4")
> my_dat2 = as.data.frame(my_dat2)
> my_dat2 = melt(my_dat2, id = "id")

# frequency table of regions ranked by choice number
> choiceFreq_regions = table(my_dat2$variable, my_dat2$value)
> choiceFreq_regions

                 Ashanti Brong Ahafo Central Eastern Greater Accra
regchoice1 31466   10210         132     244     117           103
regchoice2 31544   10316         142     160     116            70
regchoice3 31661   10235         144     158     115            56
regchoice4 31799   10180         155     128     103            53

           Northern Upper East Upper West Volta Western
regchoice1      339        165        117    15      45
regchoice2      288        163        116    16      22
regchoice3      261        158        114    23      28
regchoice4      259        146         95    14      21

# searching for the variables of expected score needed for admission
varnames[grep("score",varnames)]
# "alladmin_score1", "alladmin_score2", "alladmin_score3" and "alladmin_score4" are
retrieved.

my_dat3 = cbind(dat$anon_studentid, dat$alladmin_score1, dat$alladmin_score2,
dat$alladmin_score3, dat$alladmin_score4)
```

```
colnames(my_dat3) = c("id", "score1", "score2", "score3", "score4")
my_dat3 = as.data.frame(my_dat3)
my_dat3 = melt(my_dat3, id = c("id"))

my_dat4 = cbind(my_dat1, my_dat3)
colnames(my_dat4) = c("id", "choice_rank", "program", "id", "score_order",
"score_values")

# plotting the average, sd, the 25th and 75th quantiles by ranked choice
avg = tapply(my_dat4$score_values, my_dat4$choice_rank, mean, na.rm = TRUE)
sd = tapply(my_dat4$score_values, my_dat4$choice_rank, sd, na.rm = TRUE)
avg=as.matrix(avg)
sd=as.matrix(sd)
quant = tapply(my_dat4$score_values, my_dat4$choice_rank, quantile, na.rm = TRUE)
quant25th = cbind(c(1,2,3,4), as.matrix(c(quant$`choice1`[2], quant$`choice2`[2],
quant$`choice3`[2], quant$`choice4`[2])))
quant75th = cbind(c(1,2,3,4), as.matrix(c(quant$`choice1`[4], quant$`choice2`[4],
quant$`choice3`[4], quant$`choice4`[4])))

par(mfrow = c(2,2))
plot(avg, xlab = "choice rank", ylab = "mean score")
plot(sd, xlab = "choice rank", ylab = "std. dev. of score")
plot(quant25th[,1], quant25th[,2], xlab = "choice rank", ylab = "25th Quantile")
plot(quant75th[,1], quant75th[,2], xlab = "choice rank", ylab = "75th Quantile")

# selecting 5 ids randomly
> random5ids = trunc(runif(5, as.numeric(min(levels(my_dat4$id))),
as.numeric(max(levels(my_dat4$id)))))
> random5ids
[1] 8707 8897  828 3165  837
# finding index values of selected ids in the dataset
> id_index_vec = c(which(my_dat4$id == random5ids[1]),
                   which(my_dat4$id == random5ids[2]),
                   which(my_dat4$id == random5ids[3]),
                   which(my_dat4$id == random5ids[4]),
                   which(my_dat4$id == random5ids[5]))
# table reporting how the expected scores change by choice ranks for the selected 5
students
> table_random5 = my_dat4[id_index_vec,]
> table_random5

         id choice_rank        program   id score_order score_values
8707   8707     choice1 HOME ECONOMICS 8707      score1           37
51660  8707     choice2 HOME ECONOMICS 8707      score2           67
94613  8707     choice3 HOME ECONOMICS 8707      score3           85
137566 8707     choice4                8707      score4           79
8897   8897     choice1                8897      score1           34
51850  8897     choice2                8897      score2           60
94803  8897     choice3                8897      score3           73
137756 8897     choice4                8897      score4           56
828     828     choice1   GENERAL ARTS  828      score1           48
43781   828     choice2                 828      score2           68
86734   828     choice3                 828      score3           79
129687  828     choice4          OTHER  828      score4           55
3165   3165     choice1                3165      score1           29
46118  3165     choice2                3165      score2           38
89071  3165     choice3                3165      score3           11
132024 3165     choice4                3165      score4           35
837     837     choice1        SCIENCE  837      score1           57
43790   837     choice2        SCIENCE  837      score2           76
86743   837     choice3        SCIENCE  837      score3           90
129696  837     choice4        SCIENCE  837      score4           70

# expected score values of choice1 are the lowest for the students with ids 8707, 8897,
828 and 837.
# expected score values of choice4 are lower than those of choice3 for the students
with ids 8707, 8897, 828 and 837.

# Creating a dummy variable "reverting"
```

```
# "reverting" = 1 if the score is decreasing in choice rank.

reg_dat = dat[complete.cases(dat$alladmin_score1), ]
reg_dat = reg_dat[complete.cases(reg_dat$alladmin_score2), ]
reg_dat = reg_dat[complete.cases(reg_dat$alladmin_score3), ]
reg_dat = reg_dat[complete.cases(reg_dat$alladmin_score4), ]

reverting = vector()
for (i in 1:nrow(reg_dat)){
  if(reg_dat$alladmin_score1[i] > reg_dat$alladmin_score2[i]) {
    reverting[i] = 1
  } else {reverting[i] = 0}
}

reg_dat = cbind(reg_dat, reverting)

### Exercise 7 Probit ###

# preparing dataset to be used in probit estimation
consistent = as.numeric(reg_dat$stud_base_bece_worst > reg_dat$stud_base_bece_best)
over_confident = as.numeric(reg_dat$stud_base_bece_likely > reg_dat$stud_base_bece_best)
under_confident = as.numeric(reg_dat$stud_base_bece_worst >
reg_dat$stud_base_bece_likely)

reg_dat = cbind(reg_dat, consistent, over_confident, under_confident)

# running a probit model of reverting on measure of consistency, overconfidence,
underconfidence
# as well as gender, age, and expected education.

> myprobit = glm(reverting ~ consistent + over_confident + under_confident + GENDER +
alladmin_age + stud_base_educ_want,
                 family=binomial(link = "probit"), data = reg_dat)
> myprobit

Call:  glm(formula = reverting ~ consistent + over_confident + under_confident +
             GENDER + alladmin_age + stud_base_educ_want, family = binomial(link =
             "probit"),
           data = reg_dat)

Coefficients:
  (Intercept)           consistent       over_confident
  under_confident              GENDERF              GENDERM
-2.49127              0.20185             -0.01555              0.07223
0.19891              0.01510
alladmin_age  stud_base_educ_want
0.02447             -0.01386

Degrees of Freedom: 10067 Total (i.e. Null);  10060 Residual
(31688 observations deleted due to missingness)
Null Deviance:      3145
Residual Deviance: 3124     AIC: 3140
```