> # Jeff Hill, ECON 613 Assignment 1

>

>

> library(stringr)

> library(dplyr)

> library(StatMeasures)

> # load data

> datapath <- "/Users/admin/Documents/Econ\_613/Data/Assignment 1/"

> datstu <- read.csv(file=paste(datapath,"datstu.csv",sep=""), header=TRUE, sep=",")

> datjss <- read.csv(file=paste(datapath,"datjss.csv",sep=""), header=TRUE, sep=",")

> datsss <- read.csv(file=paste(datapath,"datsss.csv",sep=""), header=TRUE, sep=",")

>

> # Warning: the datstu$attended forloop during exercise 2 takes about 8 minutes on my computer to run.

>

> # Exercise 1

>

> # 1A Number of students

>

> # the variable "X" works as student ID, but need to check for missing values

> mean(is.na(datstu$X))

> # mean = 0, meaning no values are missing. alternatively, use unique to check for repeat IDs

> length(unique(datstu$X))

> length(datstu$X)

> # answer: 340823 students

>

> # 1B Number of schools

>

> # combining all 6 school choices of each student into one vector

> svec <- c(datstu$schoolcode1,datstu$schoolcode2,datstu$schoolcode3,datstu$schoolcode4,datstu$schoolcode5,datstu$schoolcode6)

> # ordering svec just to visualize repeated values in it

> svec <- svec[order(svec)]

> # removing repeated values with unique()

> svec <- unique(svec)

> # finally check for NA values, then take length.

> sum(is.na(svec))

> svec <- svec[!is.na(svec)]

> length(svec)

> # 640 schools applied to.

> length(unique(datsss$schoolcode))

> # this reports 898 unique school codes in datsss. HOWEVER, 209 of those schoolcodes are missing data on school name and

> # long and lat. They only possess schoolcode and district data.

> # excluding these rows with missing data there are 689 unique schools

> # Also notably in exercise 2 below, I match the each student to one of the 640 schools they applied to. All 640 school applied

> # to are included in the 689 unique schools in datss after excluding rows with missing data. This supports the conclusion that

> # the rows with missing data are not unique schools, but data that should be excluded, perhaps the product of input error.

> # answer: 689 unique schools

>

>

> # 1C number of programs

>

> # checking out choicepgm

> class(datstu$choicepgm1)

> # choicpgm columns are factors, so need to check for total amount of unique values, not counting no program declared

> length(unique(c(datstu$choicepgm1, datstu$choicepgm2, datstu$choicepgm3, datstu$choicepgm4, datstu$choicepgm5, datstu$choicepgm6)))

> print(levels(datstu$choicepgm3))

> # length is 33, however that includes the level of "not declaring a program", so the number of programs is actually 32.

> # answer: 32

>

> # 1D Number of choices (school,program)

>

> #save datstu as stu so i can manipulate datstu without losing or altering data.

> stu <- datstu

> # small for loop to create school/program columns

> for (i in 1:6) {

+ datstu[[paste("spchoice",i,sep="")]] <- paste(datstu[[paste("schoolcode",i,sep="")]], datstu[[paste("choicepgm",i,sep="")]],sep="")

+ }

> # now follow similar path as for schools above, find unique combinations.

> spvec <- c(datstu$spchoice1,datstu$spchoice2,datstu$spchoice3,datstu$spchoice4,datstu$spchoice5,datstu$spchoice6)

> spvec <- unique(spvec)

> # since we pasted two columns together into new columns, any NA values were pasted as well as characters. remove these searching for

> # pattern = "NA" using str\_detect.

> spvec <- spvec[!str\_detect(spvec, "NA")]

> # so now every element of spvec has a real character value, however some students only listed schools, and did not put in programs.

> # we want to drop all these leaving only school/program combinations. numbers\_only is a function that detects if an element is

> # only comprised of numbers.

> numbers\_only <- function(x) !grepl("\\D", x)

> spvec <- spvec[!numbers\_only(spvec)]

> length(spvec)

> # answer: 2773 unique school / program combinations

>

> # 1E Missing Test Score

>

> sum(is.na(datstu$score))

> # answer: 179887 students with missing test scores

>

> # 1F applied to the same school

>

> # This loop goes through the rows and checks if any of the school codes for each student are identical. If any are,

> # it adds one to n and then moves to the next row.

> n=0

> for (i in 1:340823) {

+ if (any(duplicated(c(datstu[i,5:10]))))

+ n <- n+1

+ }

> print(n)

> # 133668 students applied to the same school

>

> # 1G Applied to less than 6 choices

>

> # sum amount of NA's present in each student's school choices, then sum if that value is greater than zero.

> NAcount <- rowSums(is.na(datstu[,5:10]))

> sum(NAcount >0)

> # answer: 17734 students applied to less than 6 choices.

>

>

> # Exercise 2

>

> # begin creating the new data set by using spvec from exercise 1, which contained unique school/program combinations.

> spdf <- data.frame(spvec)

> # Now need to use datsss to fetch info about the high schools. however there is a lot of repeated data in datsss, so we need

> # to clean it up. create a new datsss to work with.

> sss <- datsss

> sss <- subset(sss, select = -c(X))

> sss <- na.omit(sss)

> sss<- unique(sss)

> # of note: datsss has many school codes that sss does not have, due to na.omit. As mentioned above in Q1B these schoolcodes

> # lack long/lat data, as well as school names. They also very often are close in school district name, which

> # is explained well by being the same district with school codes 1 or 2 off. They are removed, as

> # they dont include useful data. Also of note, no school names are lost using na.omit. I explore that below.

> length(unique(datsss$schoolcode))

> length(unique(sss$schoolcode))

> length(unique(datsss$schoolcode[!(datsss$schoolcode %in% sss$schoolcode)]))

> length(unique(datsss$schoolname))

> length(unique(sss$schoolname))

>

>

> # now this highschool dataset has been dramatically reduced, but there are still repeated rows that we should remove.

> # What's tricky is that many of these schools share the same long/lat, so we will need to use schoolcode to remove

> # the repeats.

> # this runs through sss$schoolcode, and removes rows where schoolcode is duplicated.

> sss<-sss[!duplicated(sss[2]),]

> # many school names include their schoolcode at the beginning, so remove those numbers for consistency.

> sss$schoolname <- gsub('[[:digit:]]+', '', sss$schoolname)

> # note: there is a single school, code = 20605 whose name was just "20605 - 20605". Those numbers were removed from the name,

> # leaving only the dash. since school names are not asked about in this assignment, I left it as is.

>

> # inserting school district into spdf (the school level dataset)

> # retrieving schoolcode and program from schoolcode/program combinations in spdf

> spdf$schoolcode <- gsub("[^0-9]", "", spdf$spvec)

> # retrieve program column and convert from character to factor

> spdf$program <- as.factor(gsub('[[:digit:]]+', '', spdf$spvec))

> # using merge, append spdf with district, long, and lat of school from sss, matching by schoolcode.

> spdf <- merge(spdf, sss[,2:5], by="schoolcode")

> # reorder the columns of spdf and rename spvec column

> spdf <- spdf[c("spvec","schoolcode","program","sssdistrict","ssslong","ssslat")]

> colnames(spdf)[colnames(spdf) == 'spvec'] <- "sppair"

> # add in cutoff

> # begin by identifying the high school each student attended by schoolcode. Add this as column to datstu

> datstu$attended <- NA

> # attended pulls the correct schoolcode, 1-6 based on rankplace. if rankplace = 99, "score too low" is the value.

> # if rankplace = NA, attended also = NA

>

> # datstu$attended <- datstu[[paste('schoolcode',datstu$rankplace,sep="")]]

> for (i in 1:340823) {

+ if (is.na(datstu$rankplace[i]) == TRUE) {

+ next

+ }

+ else if (datstu$rankplace[i] == 99) {

+ datstu$attended[i] <- 'score too low'

+ next

+ }

+ else {datstu$attended[i] <- datstu[[paste("spchoice",datstu$rankplace[i],sep="")]][i]}

+ }

> # in loop create temporary list of students based on sppair, then pull all scores.

> for (i in 1:2773) {

+ tempstu <- filter(datstu, attended == spdf[["sppair"]][i])

+ spdf$cutoff[i] <- min(tempstu[["score"]], na.rm = TRUE)

+ spdf$quality[i] <- mean(tempstu[["score"]], na.rm = TRUE)

+ spdf$size[i] <- length(tempstu[["score"]])

+ }

There were 50 or more warnings (use warnings() to see the first 50)

> sum(spdf$size ==0)

> # there are 473 missing values each in the columns cutoff, quality, and size. These were due to there being

> # 473 school / program pairs that were applied to by students, but no one got in. These could be explained

> # by possibly students applying to a school / program when that school does not offer that program. Or perhaps

> # scores were simply not good enough to get in to that school program pair, although since no one got into these

> # 473 school / program pairs, that seems less likely.

>

> # clean up the missing values in these columns by converting their cutoff and quality values to NA. size for these missing

> # obs will be left at 0.

> spdf[,7:9] <- na\_if(spdf[,7:9], Inf)

> spdf[,7:9] <- na\_if(spdf[,7:9], 'NaN')

>

> # the dataframe spdf now contains district, lat, long, cutof, quality, and size of each school / program pair. In addition,

> # it also contains school / program pair name, school code, and program name for convenience.

>

>

> # Exercise 3

>

> # begin by pulling in jss long and lat into datstu for each student, then sss long and lat for school student attended

> # then calculate distances.

> length(unique(datjss$jssdistrict))

> length(unique(datstu$jssdistrict))

> table(datstu$jssdistrict)

> # the lengths of jssdistrict in datstu and datjss are off by 1. This 1 is NA in datstu.

> # 25 students have missing jssdistrict, a very tiny portion.

> datstu <- merge(datstu, datjss[,2:4], by="jssdistrict", all=T)

> # rename point\_x and point\_y

> names(datstu)[names(datstu) == 'point\_x'] <- 'jsslong'

> names(datstu)[names(datstu) == 'point\_y'] <- 'jsslat'

> # reorder columns because merge() rearranges them

> datstu <- datstu[c("X","score","agey","male","schoolcode1","schoolcode2","schoolcode3","schoolcode4","schoolcode5","schoolcode6",

+ "choicepgm1","choicepgm2","choicepgm3","choicepgm4","choicepgm5","choicepgm6","rankplace","spchoice1",

+ "spchoice2","spchoice3","spchoice4","spchoice5","spchoice6","attended","jssdistrict","jsslong","jsslat")]

> # rename attended to sppair to match in merge

> names(datstu)[names(datstu) == 'attended'] <- 'sppair'

> # now fetch ssslong and ssslat for school attended.

> datstu <- merge(datstu, spdf[,c(1,5,6)], by="sppair", all=T)

> datstu$distance <- sqrt((69.172\*(datstu$ssslong - datstu$jsslong)\*cos(datstu$jsslat/57.3))^2 + (69.172\*(datstu$ssslat - datstu$jsslat))^2)

> # so datstu$distance now contains the distance between students' junior high and the senior high they were accepted into, if

> # they were accepted into one.

> # alternatively you could look at the distance between junior high school for each student and all 6 senior high schools they

> # potentially applied to, however the distance between the junior high and the ACTUAL school they attended seems to be more

> # valuable.

>

>

> # Exercise 4

>

> # create new data frame the has ranked choice as columns and cutoff / quality / distance avg and sd as rows.

> descha <- data.frame(matrix(ncol = 6, nrow = 6), row.names = c("cutoffavg","cutoffsd","qualityavg","qualitysd","distanceavg","distancesd"))

> colnames(descha) <- c("rankchoice1","rankchoice2","rankchoice3","rankchoice4","rankchoice5","rankchoice6")

>

> # for each student in datstu, pull in the cutoff and quality for the school they were accepted into. Then look at

> # just the students that were accepted into their first choice (rank = 1), then calculate avg and sd. do this for

> # all 6 ranks.

> datstu <- merge(datstu,spdf[,c(1,7,8)], by= "sppair", all=T)

> descha[1,] <- aggregate(datstu$cutoff, by=list(Category=datstu$rankplace), FUN=mean, na.rm = T)[1:6,2]

> descha[2,] <- aggregate(datstu$cutoff, by=list(Category=datstu$rankplace), FUN=sd, na.rm = T)[1:6,2]

> descha[3,] <- aggregate(datstu$quality, by=list(Category=datstu$rankplace), FUN=mean, na.rm = T)[1:6,2]

> descha[4,] <- aggregate(datstu$quality, by=list(Category=datstu$rankplace), FUN=sd, na.rm = T)[1:6,2]

> descha[5,] <- aggregate(datstu$distance, by=list(Category=datstu$rankplace), FUN=mean, na.rm = T)[1:6,2]

> descha[6,] <- aggregate(datstu$distance, by=list(Category=datstu$rankplace), FUN=sd, na.rm = T)[1:6,2]

> print(descha)

>

> # repeating this table differentiating by score quantiles. I am assuming quantiles means quartiles,

> # since the assignment sheet does not specify what type of quantile.

> summary(datstu$score)

> min <- summary(datstu$score)[[1]][1]

> firstq <- summary(datstu$score)[[2]][1]

> secondq <- summary(datstu$score)[[4]][1]

> thirdq <- summary(datstu$score)[[5]][1]

> max <- summary(datstu$score)[[6]][1]

>

> # Note: if a student's score fell exactly on the 1st quartile upper bound, i included that student in the 1st quartile.

> datstu$quartile[datstu$score>=min & datstu$score<=firstq]<-1

> datstu$quartile[datstu$score>firstq & datstu$score<=secondq]<-2

> datstu$quartile[datstu$score>secondq & datstu$score<=thirdq]<-3

> datstu$quartile[datstu$score>thirdq & datstu$score<=max]<-4

> datstu$quartile[is.na(datstu$score)]<- NA

>

> # create second dataframe that has cutoff / quality / distance avgs and sds for each quartile

> deschaq <- data.frame(matrix(ncol = 4, nrow = 6), row.names = c("cutoffavg","cutoffsd","qualityavg","qualitysd","distanceavg","distancesd"))

> colnames(deschaq) <- c("quartile1","quartile2","quartile3","quartile4")

>

> # fill deschaq

> deschaq[1,] <- aggregate(datstu$cutoff, by=list(Category=datstu$quartile), FUN=mean, na.rm = T)[1:4,2]

> deschaq[2,] <- aggregate(datstu$cutoff, by=list(Category=datstu$quartile), FUN=sd, na.rm = T)[1:4,2]

> deschaq[3,] <- aggregate(datstu$quality, by=list(Category=datstu$quartile), FUN=mean, na.rm = T)[1:4,2]

> deschaq[4,] <- aggregate(datstu$quality, by=list(Category=datstu$quartile), FUN=sd, na.rm = T)[1:4,2]

> deschaq[5,] <- aggregate(datstu$distance, by=list(Category=datstu$quartile), FUN=mean, na.rm = T)[1:4,2]

> deschaq[6,] <- aggregate(datstu$distance, by=list(Category=datstu$quartile), FUN=sd, na.rm = T)[1:4,2]

> print(deschaq)

>

>

> # Exercise 5

>

> # generate decile column based of cutoff of school / program pair in spdf

> spdf$decilecutoff <- decile(spdf$cutoff)

>

> # create new df select that is subset of datstu that has data we need for this exercise.

> select <- datstu[,c("sppair","X","spchoice1","spchoice2","spchoice3","spchoice4","spchoice5","spchoice6")]

> # begin merging select and spdf, pulling the corresponding decile from spdf into select based on spchoice 1 through 6

> select <-merge(select, spdf[,c(1,10)], by.x = "spchoice1", by.y = "sppair", all.x = TRUE)

> names(select)[names(select) == 'decilecutoff'] <- 'dec1'

> select <-merge(select, spdf[,c(1,10)], by.x = "spchoice2", by.y = "sppair", all.x = TRUE)

> names(select)[names(select) == 'decilecutoff'] <- 'dec2'

> select <-merge(select, spdf[,c(1,10)], by.x = "spchoice3", by.y = "sppair", all.x = TRUE)

> names(select)[names(select) == 'decilecutoff'] <- 'dec3'

> select <-merge(select, spdf[,c(1,10)], by.x = "spchoice4", by.y = "sppair", all.x = TRUE)

> names(select)[names(select) == 'decilecutoff'] <- 'dec4'

> select <-merge(select, spdf[,c(1,10)], by.x = "spchoice5", by.y = "sppair", all.x = TRUE)

> names(select)[names(select) == 'decilecutoff'] <- 'dec5'

> select <-merge(select, spdf[,c(1,10)], by.x = "spchoice6", by.y = "sppair", all.x = TRUE)

> names(select)[names(select) == 'decilecutoff'] <- 'dec6'

>

> # calculate unique amount of deciles in student's application

> # need to account for NA being unique value, where NA in decile column represents the student not having s

> # pchoice filled in, causing no decile value to be brought in by merge.

> select[["unique"]] <- apply(select[match("dec1", names(select)): match("dec6", names(select))],

+ 1, function(x) length(unique(x)))

> # however the unique column still counts NAs, so we need to subtract 1 from each value that has an NA in at least

> # one of the 6 deciles

> select[["anyNA"]] <- ifelse(rowSums(is.na(select[match("dec1", names(select))

+ :match("dec6", names(select))]))>0,1,0)

> # final step is to subtract the two columns

> select[["unique\_no\_NA"]] <- select[["unique"]] - select[["anyNA"]]

> print(select$unique\_no\_NA)

> # take students that applied to each school, take their quartile based on the test score

> # now look at how test score quartile and number of groups interact.

>

> # create selectscore, a dataframe containing our new data by quantile and by statistic.

> selectscore <- data.frame(matrix(ncol = 4, nrow = 4), row.names = c("quartile1","quartile2","quartile3","quartile4"))

> colnames(selectscore) <- c("min","mean","max","sd")

>

> # create new dataframe merging datstu with unique\_no\_NA from select, merging by X (student ID)

> merge5 <- merge(datstu, select[c("X","unique\_no\_NA")], by= "X")

>

> # fill selectscore with appropriate statistics

> selectscore[1,1:3] <-summary(merge5$unique\_no\_NA[merge5$quartile==1])[c(1,4,6)]

> selectscore[1,4] <- sd(merge5$unique\_no\_NA[merge5$quartile==1], na.rm = T)

> selectscore[2,1:3] <-summary(merge5$unique\_no\_NA[merge5$quartile==2])[c(1,4,6)]

> selectscore[2,4] <- sd(merge5$unique\_no\_NA[merge5$quartile==2], na.rm = T)

> selectscore[3,1:3] <-summary(merge5$unique\_no\_NA[merge5$quartile==3])[c(1,4,6)]

> selectscore[3,4] <- sd(merge5$unique\_no\_NA[merge5$quartile==3], na.rm = T)

> selectscore[4,1:3] <-summary(merge5$unique\_no\_NA[merge5$quartile==4])[c(1,4,6)]

> selectscore[4,4] <- sd(merge5$unique\_no\_NA[merge5$quartile==4], na.rm = T)

> print(selectscore)