```
# Exercise 1

datjss = read_csv("datjss.csv")

datsss = read_csv("datsss.csv")

datstu_v2 = read_csv("datstu_v2.csv")

# EX 1 (A)

COUNT_SCHOOL = rapply(datsss, function(x) length(unique(x)))

COUNT_SCHOOL # There are 898 schools

COUNT_STUDENT = rapply(datstu_v2, function(x) length(unique(x)))

COUNT_STUDENT # There are 340823 students

datstu = datstu_v2

programsmatrix = matrix(as.matrix(datstu[,11:16]),nrow=nrow(datstu)*6, ncol = 1)

programsmatrix = as.data.frame(programsmatrix)

COUNT_PROGRAMS = rapply(programsmatrix, function(x) length(unique(x)))

COUNT_PROGRAMS # There are 33 programs

# EX1 (B) there are 3086 choices

dat1 = gather(datstu, schchoicetype, schoolcode, schoolcode1:schoolcode6, factor_key=TRUE)

programchoice = datstu[,c(11:16)]

programchoice = gather(programchoice, prochoicetype, proname, choicepgm1:choicepgm6, factor_key=TRUE)

dat1 = dat1[,-c(5:10)]

dat1 = cbind(dat1,programchoice)

dat1 = dat1 %>% mutate(choice = paste(dat1$schoolcode,dat1$proname))

count_choice = rapply(dat1, function(x) length(unique(x)))

count_choice

# EX1 (C) There are 262146 students who applied to at least one senior school in the same district to home.

test = dat1

test = merge(test, datjss, by = "jssdistrict", all.x = TRUE)

datsss = datsss[!duplicated(datsss$schoolcode),]
```

```
test = merge(test, datsss, by = "schoolcode", all.x = TRUE)
dat1 = test
test1 = test[test$jssdistrict == test$sssdistrict,]
countq1_3 = rapply(test1 %>% select(V1.x), function(x) length(unique(x)))
countq1_3
# EX1 (D) Only 2300 choices have information about their admitted student
studentadmitted = dat1
studentadmitted$rankplace = as.numeric(dat1$rankplace)
studentadmitted = filter(studentadmitted, !is.na(studentadmitted$rankplace))
studentadmitted$schchoicetype = str_sub(studentadmitted$schchoicetype, start = 11, end = 11)
studentadmitted = studentadmitted[studentadmitted$rankplace == studentadmitted$schchoicetype,]
highschoolinfo1 = studentadmitted %>% group_by(choice) %>% summarize(size = sum(schoolcode)/schoolcode)
highschoolinfo1 = highschoolinfo1[!duplicated(highschoolinfo1$choice),]
# EX1 (E)
highschoolinfo2 = studentadmitted %>% group_by(choice) %>% summarize(cutoff = min(score))
# EX1 (F)
highschoolinfo3 = studentadmitted %>% group_by(choice) %>% summarize(quality = mean(score))


# Exercise 2
choice = merge(highschoolinfo1,highschoolinfo2,by = "choice")
choice = merge(choice, highschoolinfo3, by = "choice")
choice = left_join(choice, studentadmitted[,c(11,17,18,19,13,14)], by = "choice")
choice = choice[!duplicated(choice$choice),]


# Exercise 3
test = dat1
test = test %>% mutate(dist = sqrt((69.172*(ssslong - point_x)*cos(point_y/57.3))^2 + (69.172*(ssslat - point_y))^2))

dat1 = test


# Exercise 4

test = dat1

test = test %>% mutate(scode_rev = str_sub(dat1$schoolcode, start = 1, end = 3))

test$proname = trimws(test$proname)

test$proname = as.character(test$proname)

test = test %>% mutate(pgm_rev = proname)

test$pgm_rev[test$pgm_rev == "General Arts"] = "arts"

test$pgm_rev[test$pgm_rev == "Visual Arts"] = "arts"

test$pgm_rev[test$pgm_rev == "Business"] = "economics"

test$pgm_rev[test$pgm_rev == "Home Economics"] = "economics"

test$pgm_rev[test$pgm_rev == "General Science"] = "science"

test$pgm_rev = replace(x = test$pgm_rev, list =  !test$pgm_rev %in% c('arts', 'economics', 'science'), values =  'Other')

rapply(test, function(x) length(unique(x)))

test = test %>% mutate(choice_rev = paste(test$scode_rev,pgm_rev))

studentadmitted = test

studentadmitted$rankplace = as.numeric(studentadmitted$rankplace)

studentadmitted = filter(studentadmitted, !is.na(studentadmitted$rankplace))

studentadmitted$schchoicetype = str_sub(studentadmitted$schchoicetype, start = 11, end = 11)

studentadmitted = studentadmitted[studentadmitted$rankplace == studentadmitted$schchoicetype,]

choice_recoded_1 = studentadmitted%>% group_by(choice_rev) %>% summarize(cutoff = min(score))

choice_recoded_2 = studentadmitted%>% group_by(choice_rev) %>% summarize(quality = mean(score))

choice_recoded = merge(choice_recoded_1, choice_recoded_2, by = "choice_rev")
```

```
test = merge(test, choice_recoded, by = "choice_rev", all.x = TRUE)
dat1 = test
cleandata = dat1[,c(1,4,5,6,7,9,10,21,24,25)]
cleandata$schchoicetype = str_sub(cleandata$schchoicetype, start = 11, end = 11)
cleandata <- cleandata[order(cleandata$score, decreasing = TRUE), ]
cleandata = cleandata[c(1:120000),]

# EX5 multinomial logit model estimating effect of test score
choice1_clean = cleandata[cleandata$schchoicetype == 1,]
rapply(choice1_clean, function(x) length(unique(x)))
# I think I should delete all the rows that contain NA value
choice1_clean = choice1_clean %>% drop_na()
choice1_clean$choice_rev = as.factor(choice1_clean$choice_rev)
test = choice1_clean
test = transform(test,choiceid=as.numeric(factor(choice_rev)))
test = test[,c(11,1:10)]
choice1_clean = test
choice1_model = choice1_clean[,c(1,2,4,5,6,9,10,11)]
choice1_model = fastDummies::dummy_cols(choice1_model, select_columns = "choice_rev", remove_most_frequent_dummy = T, remove_selected_columns = T)
param = runif(length(unique(choice1_model$choiceid))*2)
like_fun = function(param,choice1_model)
{
  chid =  choice1_model$choiceid
  score      =  choice1_model$score
  age      =  choice1_model$agey
  male   =  choice1_model$male
  dist      =  choice1_model$dist
  cutoff        =  choice1_model$cutoff
  ni = nrow(choice1_model)
  nj = length(unique(choice1_model[,1]))
  ut = mat.or.vec(ni,nj)
  pn1    = param[1:nj]
  pn2    = param[(nj+1):(2*nj)]

  for (j in 1:nj)
  {
    ut[,j] = pn1[j] + score*pn2[j]
  }
  prob   = exp(ut)
  prob   = sweep(prob,MARGIN=1,FUN="/",STATS=rowSums(prob))
  probc = NULL
  for (i in 1:ni)
  {
    probc[i] = prob[i,chid[i]]
  }
  probc[probc>0.999999] = 0.999999
  probc[probc<0.000001] = 0.000001
  like = sum(log(probc))
  return(-like)
}
like_fun(param,choice1_model)

find_opt = function(ntry, choice1_model, start_val_range_a, start_val_range_b){
  out = mat.or.vec(ntry, length(unique(choice1_model)))
  lik_list = c()
  se_matrix = mat.or.vec(ntry, length(unique(choice1_model)))
  for (i0 in 1:ntry){
    i0
     start = runif(length(unique(choice1_model)), start_val_range_a, start_val_range_b)
    res = optim(start, fn=like_fun, method="BFGS", control=list(trace=0, maxit=1000), 
choice1_model = choice1_model, hessian = TRUE)
    out[i0,] = res$par
    lik_list = c(lik_list, like_fun(res$par, choice1_model))
    if (is.singular.matrix(res$hessian)){
      se_matrix[i0,] = NA
    }
    else{
      fisher_info = solve(res$hessian)
      prop_sigma = sqrt(diag(fisher_info))
      se_matrix[i0,] = prop_sigma}
  }
  out = cbind(out, se_matrix, -lik_list)
  df_out = as.data.frame(out)
  return(df_out)
}
find_opt(100, choice1_model, -1, 1)

# Ex6 Multinomial Logit Model estimating effect of school quality
like_fun_1 = function(param,choice1_model)
{
  chid =  choice1_model$choiceid
  quality       =  choice1_model$quality
  age      =  choice1_model$agey
  male   =  choice1_model$male
  dist      =  choice1_model$dist
  cutoff        =  choice1_model$cutoff
  ni = nrow(choice1_model)
  nj = length(unique(choice1_model[,1]))
  ut = mat.or.vec(ni,nj)
  pn1    = param[1:nj]
  pn2    = param[(nj+1):(2*nj)]

  for (j in 1:nj)
  {
    ut[,j] = pn1[j] + quality*pn2[j]
  }
  prob   = exp(ut)
  prob   = sweep(prob,MARGIN=1,FUN="/",STATS=rowSums(prob))
  probc = NULL
  for (i in 1:ni)
  {
    probc[i] = prob[i,chid[i]]
  }
  probc[probc>0.999999] = 0.999999
  probc[probc<0.000001] = 0.000001
  like = sum(log(probc))
  return(-like)
}
like_fun(param,choice1_model)

find_opt = function(ntry, choice1_model, start_val_range_a, start_val_range_b){
  out = mat.or.vec(ntry, length(unique(choice1_model)))
  lik_list = c()
  se_matrix = mat.or.vec(ntry, length(unique(choice1_model)))
  for (i0 in 1:ntry){
    i0
    start = runif(length(unique(choice1_model)), start_val_range_a, start_val_range_b)
    res = optim(start, fn=like_fun_1, method="BFGS", control=list(trace=0, maxit=1000), choice1_model = choice1_model, hessian = TRUE)
    out[i0,] = res$par
    lik_list = c(lik_list, like_fun_1(res$par, choice1_model))
    if (is.singular.matrix(res$hessian)){
      se_matrix[i0,] = NA
    }
    else{
      fisher_info = solve(res$hessian)
      prop_sigma = sqrt(diag(fisher_info))
      se_matrix[i0,] = prop_sigma}
  }
  out = cbind(out, se_matrix, -lik_list)
  df_out = as.data.frame(out)
  return(df_out)
}
find_opt(100, choice1_model, -1, 1)
```