Emprical Methods Project

Emine Tasci

This is a replication project of Fack, Grenet, He(2019). They use Matlab and I use R. Therefore, even though I did not use Matlab, I need to understand and learn some functions of it. Some of the Matlab functions are available in R also thanks to matlab library. I could not find substitutes for some functions, I write some basic for/if codes. This is probably because I'm not an expert in either. I feel more comfortable with R but what I learned from the project is that working with matrix and arrays would be easier in Matlab.

```
library(matlab)
```

```
##
## Attaching package: 'matlab'

## The following object is masked from 'package:stats':
##
##     reshape

## The following objects are masked from 'package:utils':
##
##     find, fix

## The following object is masked from 'package:base':
##
##     sum
```

```
library(MASS)
library(copula)
library("Rcpp")
library("matchingR")
library(IDPmisc)
library(pracma)
```

```
##
## Attaching package: 'pracma'

## The following object is masked from 'package:IDPmisc':
##
##     peaks

## The following objects are masked from 'package:copula':
##
##     polylog, psi, sinc

## The following objects are masked from 'package:matlab':
##
##     ceil, eye, factors, fliplr, flipud, hilb, isempty, isprime,
##     linspace, logspace, magic, meshgrid, mod, ndims, nextpow2, numel,
##     ones, pascal, pow2, primes, rem, repmat, rosser, rot90, size, std,
##     strcmp, tic, toc, vander, zeros
```

```
library(mgcv)
```

```
## Loading required package: nlme

## This is mgcv 1.8-34. For overview type 'help("mgcv-package")'.

##
## Attaching package: 'mgcv'
```

```
## The following object is masked from 'package:pracma':
##
##     magic
```

```
## The following object is masked from 'package:matlab':
##
##     magic
```

```
propor = 5
J= 6 #number of schools
A=4
M=100
unit_cost=1e-6

I = propor*100 #number of students
Capacities = propor * c(10,10,5,10,30,30)
rho=0.7 #correlation of student priorities across schools
Z=20 #max number of iterations

#Coefficients
FE=c(10,10.5,11,11.5,12,12.5)
coeff_score = 3
coeff_dist = -1

PARAM = list(propor=propor,I=I,J=J, Capacities=Capacities,
               A=A, M=M, unit_cost=unit_cost,Z=Z,FE=FE,
               coeff_score=coeff_score, coeff_dist=coeff_dist,rho=rho)
```

STUDENT SCORES AND PRIORITY INDICES

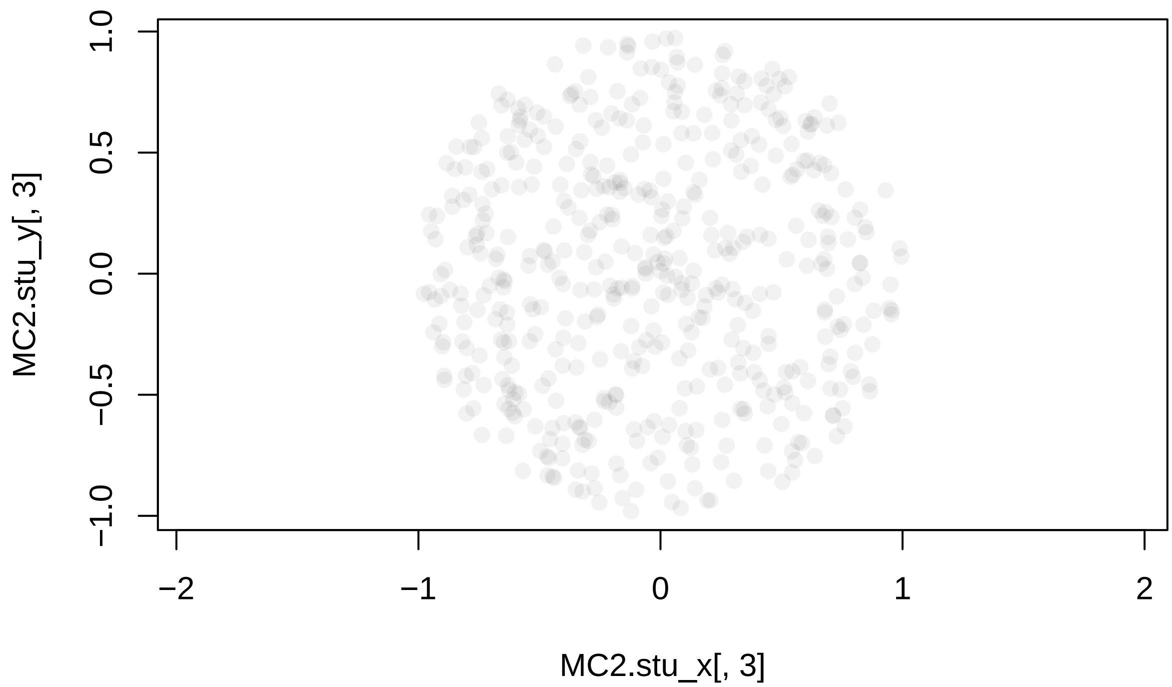
```
#MC samples #0;: M samples to compute school quality
MC0.Stu_score <- array(data=NA, dim = c(I,M))
MC0.Priorities <- array(data=NA, dim = c(J,I,M))

#MC samples #1: M new samples to compute equilibrium distribution of cutoff
MC1.Stu_score <- array(data=NA, dim = c(I,M))
MC1.Priorities <- array(data=NA, dim = c(J,I,M))

#MC samples #2: M new samples to generate simulated school choice data
MC2.Stu_score <- array(data=NA, dim = c(I,M))
MC2.Priorities <- array(data=NA, dim = c(J,I,M))






for(mm in 1:(3*M)){
  Cov_Matrix = diag(J+1) + PARAM$rho * (ones(J+1,J+1) - diag(J+1))
  #set.seed(mm)
  x2<- mvrnorm(n =I, mu=c(0,0,0,0,0,0,0), Sigma=Cov_Matrix, tol = 1e-06, empirical = FALSE)
  y <- pnorm(x2,0,1)


 if(mm <= M){
   MC0.Stu_score[,mm] <- y[,1]
   MC0.Priorities[, ,mm] <- t(y[,2:(J+1)])
```

```
 }

if(mm > M & mm <= 2*M){
    MC1.Stu_score[,mm-M] <- y[,1]
    MC1.Priorities[, ,mm-M] <- t(y[,2:(J+1)])
}


if( mm > 2*M){
    MC2.Stu_score[,mm-2*M] <- y[,1]
    MC2.Priorities[, ,mm-2*M] <- t(y[,2:(J+1)])
}
}
```

DISTANCE TO SCHOOL

```
#Schools are located on a circle of radius 1/2 and are equally spaced on that circle
MC.school_x = zeros(J,1)
MC.school_y = zeros(J,1)

for(j in 1:J){
  MC.school_x[j] = cospi(1/2 +(j-1)/3) /2
  MC.school_y[j] = sinpi(1/2 +(j-1)/3) /2
}

disc <- function(radius,angle,dim1,dim2){

}

#Students are randomly distributed on a disc of radius 1
# New coordinates are randomly drawn for each set of M Monte Carlo samples

#Student coordinates in MC samples #0
MC0.r <- sqrt(runif(I*M, min=0, max=1) ) *2 # random radius
MC0.theta <- runif(I*M, min=0, max=2*pi)    # random angle
x0 <- MC0.r * cos(MC0.theta) /2
y0 <- MC0.r * sin(MC0.theta) / 2

MC0.stu_x <- matrix(x0,nrow=500,byrow=TRUE)
MC0.stu_y <- matrix(y0,nrow=500,byrow=TRUE)

#check plot(MC0.stu_x[,3], MC0.stu_y[,3],  pch=19, col=rgb(0,0,0,0.05), asp=1)

#Student coordinates in MC samples #1
MC1.r <- sqrt(runif(I*M, min=0, max=1) ) *2 # random radius
MC1.theta <- runif(I*M, min=0, max=2*pi)    # random angle
x1 <- MC1.r * cos(MC1.theta) /2
y1 <- MC1.r * sin(MC1.theta) / 2

MC1.stu_x <- matrix(x1,nrow=500,byrow=TRUE)
MC1.stu_y <- matrix(y1,nrow=500,byrow=TRUE)
#check plot(MC1.stu_x[,3], MC1.stu_y[,3],  pch=19, col=rgb(0,0,0,0.05), asp=1)

#Student coordinates in MC samples #2
```

```r
MC2.r <- sqrt(runif(I*M, min=0, max=1) ) *2 # random radius
MC2.theta <- runif(I*M, min=0, max=2*pi)     # random angle
x2 <- MC2.r * cos(MC2.theta) /2
y2 <- MC2.r * sin(MC2.theta) / 2

MC2.stu_x <- matrix(x2,nrow=500,byrow=TRUE)
MC2.stu_y <- matrix(y2,nrow=500,byrow=TRUE)
#check
plot(MC2.stu_x[,3], MC2.stu_y[,3],  pch=19, col=rgb(0,0,0,0.05), asp=1)
```




```r
#Distance to schools
distance_fun <- function(x1,x2,y1,y2){
  d <- (x1-x2)^2 +(y1-y2)^2
  return(sqrt(d))
}


MC0.distance_school = array(data=NA, dim = c(I,J,M))
MC1.distance_school = array(data=NA, dim = c(I,J,M))
MC2.distance_school = array(data=NA, dim = c(I,J,M))

for(i in 1:I){
  for(j in 1:J){
    for(m in 1:M){
      MC0.distance_school[i,j,m] = distance_fun(MC0.stu_x[i,m], MC.school_x[j], MC0.stu_y[i,m],  MC.scho
      MC1.distance_school[i,j,m] = distance_fun(MC1.stu_x[i,m], MC.school_x[j], MC1.stu_y[i,m],  MC.scho
      MC2.distance_school[i,j,m] = distance_fun(MC2.stu_x[i,m], MC.school_x[j], MC2.stu_y[i,m],  MC.scho

    }
  }
}
```

SCHOOL QUALITY

```
#Run unconstrained DA using 100 MC preliminary samples to set the score of each school

# Random utility model : U_ij = V_ij + E_ij
# Vij : Deterministic component
# E_ij: Random component (Type-I extreme value)

rep_FE <-t(replicate(500,FE))
rep_FE <- array(rep_FE, c(I,J,M) )

MC0.V_ij_a = rep_FE + coeff_dist * MC0.distance_school

# Idiosyncratic component of utility
MC0.E_ij  = array(-log(-log(runif(I*J*M))), dim=c(I,J,M))

#Ranks = rank(MC0.Priorities[1,,1])
#View(Ranks)

MC0.Ranks <- MC0.Priorities
for(j in 1:J){
  for(m in 1:M){
   # MC0.Ranks[j, , m] <- rank(MC0.Priorities[j, ,m])
    MC0.Ranks[j, , m] <- order(-MC0.Priorities[j, ,m])

  }
}



MC.School_scores = array(data=NA, dim = c(J,100))
School_score_new =zeros(1,J)
son <- array(data=NA, dim = c(I,M))


for(m in 1:100){
  for(p in 1:100){
    if(p == 1){
      # First iteration: all school scores are set to zero
      School_score_old2 = zeros(1,J)
      School_score_old = zeros(1,J)
    }
    if (p == 2){
      # Subsequent iterations: update average school score
      School_score_old2 = zeros(1,J)
      School_score_old = School_score_new

    }
    if (p > 2){
      # Subsequent iterations: update average school score
      School_score_old2 = School_score_old
      School_score_old = School_score_new
    }
```

```
    #Deterministic component of utility
    MC0ss <- MC0.Stu_score[,m]
    Schs <- repmat(as.vector(School_score_old),I,1)

    V_ij = MC0.V_ij_a[ , ,m] +    coeff_score* Schs * replicate(J,MC0ss)

    # Utility for each school
    U_ij = V_ij + MC0.E_ij[, , m]


    #ROLs: (IxJ) matrix where (i,j) = id of school ranked j-th in student i's preferences
    # (from top = col 1 to bottom = col J)
    # [~, Submitted_ROL] = sort(U_ij,2,'descend');

    # Example:
#t <- c(9.9772107,  9.7662485,12.271464,11.9293,10.89334,11.420827)
#order(-t)  =  3 4 6 5 1 2 #this is ROL
# rank(t) = 2 1 6 5 3 4 this is st_rk_pref


    Submitted_ROL <- U_ij

      for(i in 1:I){
        Submitted_ROL[i, ] <- order(-U_ij[i, ])
      }


    #Students' preferences over schools:
    #(I,J) matrix where (i,j) is the preference of student i for school j,
    # in ranks (from 1 for least preferred to J for most preferred)
    Stu_rk_pref <- U_ij

      for(i in 1:I){
        Stu_rk_pref[i, ] <- rank(U_ij[i, ])
      }




  #Run SPDA
  SPDA<-  galeShapley.collegeAdmissions(  studentPref = t(Submitted_ROL)  ,
      collegePref = t(MC0.Ranks[,,m]) ,
      slots = Capacities,
      studentOptimal = TRUE )


  # Determine average score of students assigned to each school
  School_score_new = array(data=NA, dim = c(1,J))

  son[,m] <- SPDA$matched.students

  data<- cbind.data.frame(son[,m],MC0.Stu_score[,m])
  names(data) <- c("dx", "dy")
```

```
  data <- NaRV.omit(data)
  for(j in 1:J){
    School_score_new[,j] <-   mean(data$dy[data$dx==j])
  }


# son[is.na(son[,m])] <- 0


#  for(j in 1:J){
#   num<-0
#  sum <- 0
#    for(i in 1:I){
#      if(son[i,m]==j){
#        num <- num +1
#        sum <- sum+ MC0.Stu_score[i,m]
#      }
#    }
#    School_score_new[,j] <- sum/num
#  }

if (School_score_new==School_score_old || School_score_new==School_score_old2 ){
    MC.School_scores[,m]=t(School_score_new)
    break
}


  }
}


scores <- as.data.frame(t(MC.School_scores))
scores <- NaRV.omit(scores)
MC.School_mscores <- colMeans(scores)
```

STUDENT PREFERENCES

```
MC1.V_ij = rep_FE + coeff_dist*MC1.distance_school +
  coeff_score*array(t(replicate(100,MC.School_mscores)), c(I,J,M) )  * aperm(replicate(J,MC1.Stu_score)

MC2.V_ij = rep_FE + coeff_dist*MC2.distance_school +
  coeff_score*array(t(replicate(100,MC.School_mscores)), c(I,J,M) )  * aperm(replicate(J,MC2.Stu_score)


MC1.E_ij  = array(-log(-log(runif(I*J*M))), dim=c(I,J,M))

MC2.E_ij = array(-log(-log(runif(I*J*M))), dim=c(I,J,M))

MC1.U_ij = MC1.V_ij + MC1.E_ij

# Alternative structure (for compatibility with parfor)
MC1_U_ij = aperm(MC1.U_ij, c(3, 2 ,1))

MC2.U_ij = MC2.V_ij + MC2.E_ij
```

```r
# Alternative structure (for compatibility with parfor)
MC2_U_ij = aperm(MC2.U_ij, c(3, 2 ,1))
```

UNCONSTRAINED CHOICE SIMULATIONS

```r
DA_UNC1.ROL <- MC1.U_ij

for(m in 1:M){
  for(i in 1:I){
    DA_UNC1.ROL[i, ,m] <- order(-MC1.U_ij[i, ,m])
  }
}


DA_UNC2.ROL <- MC1.U_ij

for(m in 1:M){
  for(i in 1:I){
    DA_UNC2.ROL[i, ,m] <- order(-MC2.U_ij[i, ,m])
  }
}



DA_UNC1.Ranks <-  MC1.Priorities
for(j in 1:J){
  for(m in 1:M){
    DA_UNC1.Ranks[j, , m] <- order(-MC1.Priorities[j, ,m])

  }
}

DA_UNC2.Ranks <-  MC2.Priorities
for(j in 1:J){
  for(m in 1:M){
    DA_UNC2.Ranks[j, , m] <- order(-MC2.Priorities[j, ,m])

  }
}


#Run SPDA to calculate school cutoffs
DA_UNC1.Cutoffs = zeros(J,M)
DA_UNC1.Assigned = zeros(M,I)
DA_UNC1.Matched <- array(data=0, dim = c(M,J,I))

DA_UNC2.Cutoffs = zeros(J,M)
DA_UNC2.Assigned = zeros(M,I)
DA_UNC2.Matched <- array(data=0, dim = c(M,J,I))


for(m in 1:M){
  SPDA1 <- galeShapley.collegeAdmissions(  studentPref = t(DA_UNC1.ROL[,,m])  ,
                                           collegePref = t(DA_UNC1.Ranks[,,m]) ,
```

```
                                                  slots = Capacities,
                                                  studentOptimal = TRUE )

  SPDA2 <- galeShapley.collegeAdmissions(  studentPref = t(DA_UNC2.ROL[,,m])  ,
                                                  collegePref = t(DA_UNC2.Ranks[,,m]) ,
                                                  slots = Capacities,
                                                  studentOptimal = TRUE )

  Cutoffs1=matrix(NA, c(J,1))
  Cutoffs2=matrix(NA, c(J,1))

  matched1 <- SPDA1$matched.colleges
  assigned1 <- SPDA1$matched.students

  matched2 <- SPDA2$matched.colleges
  assigned2 <- SPDA2$matched.students


  data1<- cbind.data.frame(assigned1,t(MC1.Priorities[,,m]))
  data1 <- NaRV.omit(data1)

  data2<- cbind.data.frame(assigned2,t(MC2.Priorities[,,m]))
  data2 <- NaRV.omit(data2)

  for(j in 1:J){
    Cutoffs1[j,1] <-   min(data1[,j+1][data1[,1]==j])
    Cutoffs2[j,1] <-   min(data2[,j+1][data2[,1]==j])

    if( length(matched1[[j]]) < Capacities[j]){Cutoffs1[j,1]=0}

    if(Cutoffs1[j,1]==min(MC1.Priorities[j,,m])){Cutoffs1[j,1]=0}

    if( length(matched2[[j]]) < Capacities[j]){Cutoffs2[j,1]=0}

    if(Cutoffs2[j,1]==min(MC2.Priorities[j,,m])){Cutoffs2[j,1]=0}
  }

#  DA_UNC1.Assigned[m,] = assigned1
#  DA_UNC1.Matched[m,,] = matched1
  DA_UNC1.Cutoffs[,m] = Cutoffs1
#  DA_UNC2.Assigned[m,] = assigned2
#  DA_UNC2.Matched[m,,] = matched2
  DA_UNC2.Cutoffs[,m] = Cutoffs2
}

#School cutoffs in MC sample 1:
rowMeans(DA_UNC1.Cutoffs)
```

```
## [1] 0.1046439 0.2071042 0.5377565 0.4398544 0.2018981 0.3772714
```

```
#In paper: 0.1034, 0.1926, 0.6724, 0.5547, 0.2116, 0.4189
```

CONSTRAINED CHOICE SIMULATIONS

FIND ALL POTENTIAL CONSTRAINED ROLs

```r
# Assumption : students can submit a maximum of A schools out of J
# K: number of rank-preserving ordering of A schools among J
# K = J!/(A!(J-A)!)  (e.g., J=5 & A=3 -> K=10)

# If application cost is zeo, only consider lists of size A


if (PARAM$unit_cost == 0){
 # included: Matrix of dummy variables that indicates the K possible ROLs of A schools
  included = t(uniquecombs(perms(cbind(ones(1,A), zeros(1,J-A)) ),ordered=FALSE))
  #Number of combinations for each student
  K= nchoosek(J,A)
}


#If application cost is strictly positive, consider all possible combination of schools
#of size 1 to A
if (PARAM$unit_cost > 0){
  included= matrix(0,nrow=6, ncol=0)

  for(a in 1:A){
  included_new= t(uniquecombs(perms(cbind(ones(1,a), zeros(1,J-a)) ),ordered=FALSE))
  included= cbind(included,included_new)
  }
  #Number of combinations for each student
  K = ncol(included)
}

PARAM$K=K

#Replace the INCLUDED matrix of 0 and 1s by the corresponding school positions of included schools
included_position= matrix(0,nrow=A, ncol=ncol(included))

for(c in 1:ncol(included)){
  n=0
  for(r in 1:nrow(included)){

    if(included[r,c]>0){
      n=n+1
      included_position[n,c]=r
      }
  }
}
```

COMPLETE ROL (TRUE PREFERENCES)

FIND BAYES-NASH EQUILIBRIUM OF CONSTRAINED SCHOOL CHOICE GAME

The last part is not finished yet. I need to find BNE, then the next step is estimation. Since it is not finished it is not here but in the .R file.