Econ 21410 - Problem Set III

Marriage Matching[*]

April 21, 2014

This homework should be done in LaTeX The homework will be graded on correctness, but will also heavily weight clarity and professionalism. Its better to not do some of the harder parts than to turn in an incomprehensible document. Your R script as well as a log-file should be submitted. Alternatively, use knitr to print your code inline in your latex document.

SUBMISSION: The homework must be emailed to Oliver and myself by 2p.m. Monday, April the 21st. The email must include a pdf with the filename lastname_pset2.pdf and R code called lastname_pset2_code.R where "lastname" should be replaced with your last name. The subject of your email should be [ECON 21410: pset3 submission]

If you are struggling, please use the github page to ask for help!* Remember that asking and answering questions on our github page, coming to office hours to ask questions, and contributing to the class wiki are all worth participation credit, which is 10% of your grade in this class.

Becker's Marriage Market Warm Up

1. Suppose the output of a marriage is determined by the function $h(m_i, f_j)$, where m_i and f_i are the amount of skill man i and woman j bring to the marriage respectively. For each of the functions below, answer if the function will lead to positive assortitive matching in (1) the transferrable utility case and (2) the non-transferrable utility case.

   - Transferable Utility: Becker (1974) showed that under transferable utility there will be transfers in equilbrium such that the sum of all individuals utility is maximized. This will imply assortative matching if $\frac{\partial^2 h(m,f)}{\partial m \partial f} > 0$ since in this the difference in productivity between a high type male and a low type male will be larger when matched with a high type female than a low type. If $\frac{\partial^2 h(m,f}{\partial m \partial f} < 0$ then non-assortative matching is optimal.

   - Non-Transferable Utility: In the non transferable case we assume that the two partners split the total utility by some pre-specified shares α and $1 - \alpha$. Now the person each individual wants to marry will depend on the sign of the partial derivatives. If

2. Suppose the output of a marriage is determined by the function $h(m_i, f_j)$, where m_i and f_i are the amount of skill man i and woman j bring to the marriage respectively. For each of the functions below, answer if the function will lead to positive assortitive matching in (1) the transferable utility case and (2) the non-transferable utility case.

[*]Please email johneric@uchicago.edu and obrowne@uchicago.edu if you have questions.

- $h(m,f) = m^{0.3}f^{0.3}$
  - Transferable Utility:Positive Sorting
  - Non-Transferable Utility: Positive Sorting
- $h(m,f) = (m+f)^2$
  - Transferable Utility: Positive Sorting
  - Non-Transferable Utility: Positive Sorting
- $h(m,f) = (m+f)^{0.5}$
  - Transferable Utility: Negative Sorting
  - Non-Transferable Utility: Positive Sorting
- $h(m,f) = m + f$
  - Transferable Utility: Any sorting pattern optimal since in equilbrium each individual will recieve their value m or f
  - Non-Transferable Utility: Positive Sorting
- $h(m,f) = min\{m,f\}$
  - Transferable Utility:
    * Typically we positive sorting because a Leonteif production function is a limit of a sequence of CES production functions
    * However it is also possible to construct examples where any sorting pattern is possible (for example if all of the women are strictly better than all of the men, then the same output will be produced regardless of the sorting pattern)
  - Non-Transferable Utility: Positive Sorting

Becker's Marriage Market Warm up

1. Write out (in words) the steps for an algorithm that calculates the division of marital output in a marriage market with more women than men, men propose to women, and the output of the marriage is super-modular (so we have positive assortitive mating).[1]

   - Rank both women and men from highest to lowest. There is positive assortitive mating, so for $i = 1, \ldots, nM$ the i^{th} ranked man will marry the i^{th} ranked woman. for $i = nM, \ldots, nF$ the i^{th} woman will remain umarried.
   - Start with the nM^{th} couple.

[1]Hint: We did this in class!

- The nM^{th} woman will recieve zero surplus since her outside option is to remain unmarried and recieve zero. $S^f_{nM} = 0$
- The nM^{th} man will receive all the match surplus $S^m_{nM} = h(m_nM, f_nM)$
- Then iterate backwards until you reach the first couple:
- The i^{th} ranked man will make a proposal to the i^{th} ranked woman which will leave here indifferent between marrying the i^{th} man and the $i+1^{th}$ man:
  $S^f_i = h(m_{i+1}, f_i) - S^m_{i+1}$
- The i^{th} man will receive what is left over from his match with the woman:
  $S^m_i = h(m_i, f_i) - S^f_i$.

2. Write out how this algorithm would change if there were more men than women, but men still proposed.

   - The algorithm would work in the same way but we would start with the nF^{th} woman.
   - The proposal of the nF^{th} man would have to offer her the same surplus as she could produce with the $nF+1^{th}$ man
     $S^f_{nF} = h(nF+1, nF)$
   - The $nF^{+}1^{th}$ man would reseive the remainded of the match surplus
     $S^m_{nF} = h(nF, nF) - S^f_{nF}$
   - Then iterate backwards as before.

3. Assume that there are more fs than ms and that m's "propose" in this model.[2] Assume the utility of not marrying is 0. In class we showed that such a setup will have positive assortitive mating. Who will women i match with if i is less than the total number men?

   The i^{th} Woman will match with the i^{th} Man.

4. Write a function that takes the "males" and "females" matrices defined below and calculates: (1) the output of each match (we already know from positive assortitive mating which male and female will match with each other) and (2) the division of the output between men and women. The function should fill in the columns of the "males" and "females" matrix and return those matrices in a list.
   See code below.

5. What proportion of the output do fs get when education has the discrete binomial distribution? Run the model a few times and make sure your initial run is not an outlier.

   Under a discrete binomial distribution females get around 25% of output.

6. Change "males" and "females" to have education levels drawn from the uniform distribution (currently commented out in the code below). How does this change the proportion of the output that the fs get on average. Run the model a few times and make sure your initial run is not an outlier.
   Under a uniform distribution females get around 37% of output.

[2]This means men propose a division of the marriage output which women can accept or reject.

7. Discuss the differences between your results in the previous two questions. Explain the economics behind why they differ.
   When Males propose they will always offer female the output equal to their opportunity cost. When we have the binomially distributed output there are discrete levels of education, and so there are often overlapping levels of education. When there is an overlap in the level of education the individual with that overlap unable to extract any extra surplus from that match above his outside option. So under a binomial distribution there will be a more skewed distribution of surplus than under a uniform distribution where since there are not discrete levels both individuals are able to extract some additional surplus at every level.

8. (if you are struggling with the problem set, skip the remaining two parts of this problem as they will be worth fewer points than the rest of the problem set.)

9. Extend your function to work in the case where there are more men than women, but men still propose.
   See code below.

10. What proportion of the output do fs now get when education is binomially distributed? How about when education is distributed uniformly? How does this differ from your result when there were fewer men than women.
    See table below. When there more males than females, females recieve around 74% of output under the Binomial distribution and around 60% of output under a uniform distribution. Again these differences in these shares occur due to the same effects from overlapping levels of education. However now the benefits of this one sided extraction fall largely on the Females. This is because there are more males so the bottom female can extract more surplus and the bottom male less. Since the amount of surplus all other females can extract is cumulative, the females in this models extract more total surplus than the males.

	Avg Female Share of Output	Std Dev
Binomial Distribution, #Female>#Male	25%	3%
Uniform Distribution, #Female>#Male	37%	4%
Binomial Distribution, #Male>#Female	74%	3%
Uniform Distribution, #Male>#Female	60%	4%

Table 1: Female Share of Output, Across 50 Simulations

```r
# Generating Agents with education for Becker
# Marriage model.  ================================

# Create Matrix Structure for Output
n <- 120
data.matrix <- matrix(0, n, 4)  # data for males to fill in
data.matrix[, 1] <- c(1:n)
colnames(data.matrix) <- c("id", "educ", "output",
    "surplus")

# Simulation 1 Binomial Distribution, More Females
# than Males
nMales1 <- 100  #number of males
nFemales1 <- 110  #number of females
```

```r
males1 <- data.matrix[1:nMales1, ]  #create data matrix
females1 <- data.matrix[1:nFemales1, ]
# generate distributions of education levels
males1[, 2] <- sort(rbinom(nMales1, 16, 0.5), decreasing = T)
females1[, 2] <- sort(rbinom(nFemales1, 16, 0.5), decreasing = T)

# Simulation 2 Uniform Distribution, More Females
# than Males
nMales2 <- 100
nFemales2 <- 110
males2 <- data.matrix[1:nMales2, ]
females2 <- data.matrix[1:nFemales2, ]
males2[, 2] <- sort(runif(nMales2, min = 0, max = 16),
    decreasing = T)
females2[, 2] <- sort(runif(nFemales2, min = 0, max = 16),
    decreasing = T)

# Simulation 3 Binomial Distribution, More Males
# than Females
nMales3 <- 110
nFemales3 <- 100
males3 <- data.matrix[1:nMales3, ]
females3 <- data.matrix[1:nFemales3, ]
males3[, 2] <- sort(rbinom(nMales3, 16, 0.5), decreasing = T)
females3[, 2] <- sort(rbinom(nFemales3, 16, 0.5), decreasing = T)

# Simulation 4 Uniform Distribution, More Females
# than Males
nMales4 <- 110
nFemales4 <- 100
males4 <- data.matrix[1:nMales4, ]
females4 <- data.matrix[1:nFemales4, ]
males4[, 2] <- sort(runif(nMales4, min = 0, max = 16),
    decreasing = T)
females4[, 2] <- sort(runif(nFemales4, min = 0, max = 16),
    decreasing = T)

# ================================
```

```r
# Section 2: Becker Matching Algorithm
#================================



output = function(wom,man=1,males,females)
{
# A function defining the output of a marriage
    out = males[man,"educ"] * females[wom,"educ"]
    return(out)
}
```

```r
BeckerMatch <- function(males=males1,females=females1,nMales,nFemales){
    # Calculates becker marriage match under the following assumptions:
    # 1.) Men Propose 2.) The output function leads to positive assortitive matching
    #
    # Inputs:
    #   nMales and nFemales: are the number of males and females respectively
    #   males,females: are (nMales x 4) and (nFemales x 4) matricies respectively where:
    #               the row "id"   gives a unique id number of the individual
    #               the row "educ" gives the match quality of an individual
    #               the rows "output" and "surplus" are completed by the function
    #
    # Outputs:
    #  a list containing the completed 'males' and 'females' matricies with the
    #  "output" and "surplus" columns completed

    for (m in nMales:1) #Loop over all males
    {
        if (nMales <= nFemales) #If fewer males than females
        {
            if (m == nMales) #If considering last male
            {
                #Generate Match Output
                males[m,"output"]   = output(wom=m,man=m,males,females)
                females[m,"output"] = males[m,"output"]
                #Male takes entire match output
                males[m,"surplus"]   = males[m,"output"]
                #Female gets zero surplus
                females[m,"surplus"] = 0
            }
            if (m < nMales) #If not considering last male
            {
                #Generate match output
                males[m,"output"]     = output(wom=m,man=m,males,females)
                females[m,"output"]   = males[m,"output"]
                #Calculate female's outside option
                secondbest_fem = output(wom=m,man=(m+1),males,females) - males[m+1,"surplus"
                #Male takes output less outside option
                males[m,"surplus"]     = males[m,"output"] - secondbest_fem
                #Female gets outside option
                females[m,"surplus"]   = females[m,"output"] - males[m,"surplus"]
            }
        }
        if (nMales > nFemales) #If more males than females
        {
            #Unmarried males get zero
            if (m>nFemales) males[m,c("output","surplus")] = c(0,0)
            if (m==nFemales) #If the last married male
            {
                #Generate match output
                males[m,"output"]     = output(wom=m,man=m,males,females)
                females[m,"output"]   = males[m,"output"]
                #Calculate outside option
                secondbest_fem = output(wom=m,man=(m+1),males,females)
                #Male gets output less outside option
                males[m,"surplus"]      = males[m,"output"] - secondbest_fem
                #Female gets outside option
                females[m,"surplus"]    = females[m,"output"] - males[m,"surplus"]
            }
            if (m<nFemales) #If not considering last male
            {
                #Generate match output
                males[m,"output"]      = output(wom=m,man=m,males,females)
                females[m,"output"]    = males[m,"output"]
                #Calculate outside option
                secondbest_fem = output(wom=m,man=(m+1),males,females) - males[m+1,"surplus"
                #Male gets output less outside option
                males[m,"surplus"]      = males[m,"output"] - secondbest_fem
                #Female gets outside option
                females[m,"surplus"]    = females[m,"output"] - males[m,"surplus"]
            }
        }
    }
    #Return data for males and females in list
    return(list(males = males,females = females))
}

#================================
```

```
# Section 3: Becker Simulation and Output Tables
# ================================

num.sim <- 50  #Number of Simulations

# Matrix for outputting the female share of each
# simulation
share.f <- matrix(rep(NA, 4 * num.sim), nrow = 4)

# Loop runs simulation num.sim times
for (i in 1:num.sim) {

    # Simulation 1 Randomly Generate Male Education
    # Levels
    males1[, "educ"] <- sort(rbinom(nMales1, 16, 0.5),
        decreasing = T)
    females1[, "educ"] <- sort(rbinom(nFemales1, 16,
        0.5), decreasing = T)
    # Find Beckerian Match Outputs
    matches1 <- BeckerMatch(males1, females1, nMales1,
        nFemales1)
    # Extract Female Matches
    females1 <- matches1[[2]]
```

```r
    # Calculate Average Female Surplus
    share.f[1, i] <- mean(females1[1:min(nMales1, nFemales1),
        "surplus"]/females1[1:min(nMales1, nFemales1),
        "output"])

    # Simulation 2
    males2[, "educ"] <- sort(runif(nMales2, min = 0,
        max = 16), decreasing = T)
    females2[, "educ"] <- sort(runif(nFemales2, min = 0,
        max = 16), decreasing = T)
    matches2 <- BeckerMatch(males2, females2, nMales2,
        nFemales2)
    females2 <- matches2[[2]]
    share.f[2, i] <- mean(females2[1:min(nMales2, nFemales2),
        "surplus"]/females2[1:min(nMales2, nFemales2),
        "output"])

    # Simulation 3
    males3[, "educ"] <- sort(rbinom(nMales3, 16, 0.5),
        decreasing = T)
    females3[, "educ"] <- sort(rbinom(nFemales3, 16,
        0.5), decreasing = T)
    matches3 <- BeckerMatch(males3, females3, nMales3,
        nFemales3)
    females3 <- matches3[[2]]
    share.f[3, i] <- mean(females3[1:min(nMales3, nFemales3),
        "surplus"]/females3[1:min(nMales3, nFemales3),
        "output"])

    # Simulation 4
    males4[, "educ"] <- sort(runif(nMales4, min = 0,
        max = 16), decreasing = T)
    females4[, "educ"] <- sort(runif(nFemales4, min = 0,
        max = 16), decreasing = T)
    matches4 <- BeckerMatch(males4, females4, nMales4,
        nFemales4)
    females4 <- matches4[[2]]
    share.f[4, i] <- mean(females4[1:min(nMales4, nFemales4),
        "surplus"]/females4[1:min(nMales4, nFemales4),
        "output"])
}

# Generate output table showing mean and SD of
# average female share across simulations
outtable <- cbind(round(100 * apply(share.f, 1, mean)),
    round(100 * apply(share.f, 1, sd)))
rownames(outtable) <- c("Binomial Distribution, #Female>#Male",
    "Uniform Distribution, #Female>#Male", "Binomial Distribution, #Male>#Female",
    "Uniform Distribution, #Male>#Female")
colnames(outtable) <- c("Average Female share of Output",
    "std.dev")
```

```r
# Generate Latex table w xtable
xtable(outtable, caption = c("Female share of output, over 20 simulations"))

# ========================
```

Gale Shapley

In the code below, I create a list of preference rankings for each man and woman in two different marriage markets. In “males1” and “females1”, preferences over members of the opposite sex are random. In “males2” and “females2” there is assortitive matching where a lower id number is strictly preferred by everyone to a higher id number.

1. Write out the steps of the Gale-Shapley algorithm

   - While matches matches are not yet stable (matches are not stable if some match changed between this and the previous iteration)
   - All unengaged men propose to the top ranked women they have not been rejected by
   - All women with multiple proposals from this and the previous iterations choose their top ranked man
   - This man gets engaged with this women
   - All other men who proposed to or were previously engaged with this woman become single again
   - End while loop

2. Implement the Gale-Shapley algorithm. Write an algorithm that takes a matrix of men’s rankings of women and a matrix of women’s ranking of men as inputs. Have this function implement the Gale-Shapely algorithm and return the final “match matrix” MM which contains a 1 in cell $MM[i, j]$ if male i marries female j and contains a 0 otherwise.

```
# Section 4: Gale-Shapley Algorithm
#========================

DeferredAcceptanceAlgorithm <- function(males, females){
# Runs a males proposing Gale-Shapley Deferred Acceptance Algorithm
#
# Inputs: males and females are (n x m) and (m x n) matricies indexed by row numbers
#   where each row describes the rank order preferences over all individuals of the oth
#
# Outputs: matches is a binary (n x m) matrix
#                    with entrys of 1 if the ith man matched with the jth woman
#                     and entrys of 0 otherwise

    matches = matrix(0,nMales,nFemales)
    prev_matches = matrix(1,nMales,nFemales)

    while (all((matches==prev_matches))==F) #Iterates until matches are stable
    {
        prev_matches = matches              #Saves previous matches
        for (m in 1:nMales)                 #Loops over all males
        {
            for (mate in order(males[m,]))  #Loops over mates in order of preference
            {
                # if neither are engaged
                if (sum(matches[m,])==0 &  sum(matches[,mate])==0){
                    matches[m,mate]=1   # They get matched
                }
                if (sum(matches[m,])==0 &  sum(matches[,mate])>0) # if woman is engage
                {
                    # identify her current fiance's index
                    otherguy = match(1,matches[,mate])
                    # check  if proposal is better than her current match
                    if (females[mate,m]   < females[mate,otherguy])
                    {
                        matches[otherguy,mate] = 0  # If so other guy gets dumped
                        matches[m,mate] = 1         # And current guy gets matched
                    }
                }
            }
        }
    }
    return(matches) # Return matches
}

#========================
```

3. Run the algorithm on the two populations below and print the output to screen. Make sure you get the correct result for the model with positive assortitive matching.

```
# Section 5: Gale-Shapley Simulation and Output
# Tables ========================

nMales <- 10
nFemales <- 20

# rankings... assumed lower is better (1st place,
# etc.)
males1 <- matrix(replicate(nMales, sample(nFemales)),
    nMales, nFemales, byrow = T)
females1 <- matrix(replicate(nFemales, sample(nMales)),
    nFemales, nMales, byrow = T)
# Find Matches using Deferred Acceptance Algorithm
matches1 <- DeferredAcceptanceAlgorithm(males1, females1)
# Output matches into Latex Table
xtable(matches1, display = rep("d", 21), caption = "Matches from randomly drawn prefere

# Preferences from Becker's Marriage model (to see
# that we get the same results)
males2 <- matrix(replicate(nMales, c(1:nFemales)),
    nMales, nFemales, byrow = T)
females2 <- matrix(replicate(nFemales, c(1:nMales)),
    nFemales, nMales, byrow = T)
# Find Matches using Deferred Acceptance Algorithm
matches2 <- DeferredAcceptanceAlgorithm(males2, females2)
# Output matches into Latex Table
```

```
xtable(matches2, display = rep("d", 21), caption = "Matches from assortative preference
```

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
3	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
7	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

Table 2: Matches from randomly drawn preferences

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

Table 3: Matches from assortative preferences

Research

In 1-3 sentences propose a research idea related to the marriage market.

Name two cases where matching models may be applicable outside of the marriage market.

Side Projects

- Side projects from previous homeworks are still valid (unless already completed, such as making a specific entry on the class wiki.)
- Read Prof Becker's original 1974 paper on this subject and write a short (1-3 page) summary and response (2.5 points).
- Rewrite some or all of the code above in Julia or C++ (with Rcpp) (up to 3.5 points).
- Read and review two applied papers which test or extend Becker's marriage model (1-3 pages, 2.5 points).

- Look for a paper which models how couples negotiate the division of output within a marriage and write a brief summary (1-2 pages, up to 2 points)
- Look for papers which discuss how divorce laws changed bargaining power. Read one and write a brief summary and response (1-2 pages, 2 points).
- Go to ipums.org and explore the variables in the latest wave of the American Community Survey (ACS) or the Consumer Population Survey (CPS). Find variables that are interesting or surprising and write up a 1-page report (up to 2 points).
- Have a research idea? Write a document that provides a (1) two sentence statement of the idea and (2) a more complete 0.3-1 page description of the idea (up to 3 points).