# Econ 21410 - Problem Set I

**Marriage Matching And Getting Started***

April 9, 2015

This homework should be done in LaTeX The homework will be graded on correctness, but will also heavily weight clarity and professionalism. Being able to produce clean, clear, well documented write-ups and code is an important skill which will be rewarded. 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. There are sample knitr documents on the course website as well as an introductory guide posted on the class wiki repository on github.com.

Please make sure you have access to github.com/CompEcon as soon as possible. If you do not have access, please email us and let us know.

Make sure to write code which is clear and flexible. Read the whole problem before you begin coding. Some parameters will change and the code should be written in a way to make this easy to implement. We will re-use code in this course. Flexibility and documentation now will save you headaches later in the quarter.

SUBMISSION: The homework must be emailed to Oliver and myself by Monday 9:30am Monday, April 6th. The email must include a pdf with the filename lastname_pset1.pdf and R code called lastname_pset1_code.R where "lastname" should be replaced with your last name. The subject of your email should be "[ECON 21410: pset1 submission]" (including the brackets).

## 1 Getting started with Github

1. You should have already made a github.com account and shared your user name with Oliver and myself, if not, do so as soon as you read this.
2. Go to "CompEcon" at github.com/CompEcon. You should be able to see 3 repositories. Go into econ21410wiki this contains a guide for using knitr made by a student last year, but the important parts are in the "Issues" and "Wiki" likes on the right hand side of the screen.
    - Go to the wiki and find the "Class Email List" page. Modify this page to add your name, email, and github handle (using my entry as an example).
    - Go to the issues tracker and see the example issue I have opened. Go back to github.com/CompEcon/econ21410wiki/issues and click on the "closed" button. These are the 91 questions/comments/issues raised by students last year. These can be used as a resource and an example on how to use the issue tracker.

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

## 2 Getting Started with R

Display this output in your code (preferrably inline with knitr). None of these should require more than a single line of R. These exercises must be calculated in R, not done by hand.

1. Print inline "hello world"
2. Create a vector $y = \begin{bmatrix} 100 \\ 200 \\ 300 \\ 400 \\ 500 \end{bmatrix}$
3. Create a matrix $X$ which is $5 \times 5$ and contains random draws from a normal with mean 100 and variance 10.
4. Calculate and display $(X'X)^{-1}$
5. Calculate the sum of the entries in $y$
6. Calculate the row sums of the entries of $X$
7. Return the maximum value in $X$
8. Replace the third row of $X$ with 0s and display it

```
#========================
# Getting Started with R, Solutions
#========================

#1)
print('hello world')


## [1] "hello world"


#2)
y <- c(100,200,300,400,500)
#3)
x <- matrix(rnorm(5*5),nrow=5)
#4)
solve(t(x) %*% x)


##           [,1]       [,2]       [,3]      [,4]
## [1,]  2.576992 -1.1282587  1.8919645  5.049085
## [2,] -1.128259  1.2915719 -0.7011866 -3.521472
## [3,]  1.891964 -0.7011866  1.5926393  3.761415
## [4,]  5.049085 -3.5214722  3.7614154 12.978501
## [5,] -3.173186  2.3727967 -2.4007508 -8.445115
##           [,5]
## [1,] -3.173186
```

```
## [2,]  2.372797
## [3,] -2.400751
## [4,] -8.445115
## [5,]  5.732792
```

```
#5)
sum(y)
```

```
## [1] 1500
```

```
#6)
rowSums(x)
```

```
## [1] -3.2645009  1.4126497 -0.4406094  0.9972629
## [5]  0.5186975
```

```
#7)
x[3,] <- rep(0,5)
x
```

```
##            [,1]       [,2]       [,3]       [,4]
## [1,]  0.5414674 -0.3189137 -0.8679182 -1.0112450
## [2,]  0.1524588  2.3673647 -1.6362082  0.9471436
## [3,]  0.0000000  0.0000000  0.0000000  0.0000000
## [4,] -1.0241894 -0.8296063  0.2586865  1.1101111
## [5,]  1.8360157  0.6968234 -2.4421829  0.2516776
##            [,5]
## [1,] -1.6078914
## [2,] -0.4181091
## [3,]  0.0000000
## [4,]  1.4822610
## [5,]  0.1763636
```

## 3 Function and Loops in R

1. Using a for loop print all numbers between 1 and 100 which are not multiples of 3 or 4

2. Write a function which takes a number as an input and returns a vector containing all numbers in the fibbonaci sequence less than that number. Use this function to print all fibbonaci numbers less than 1000.

```
#=======================
# Functions and loops in R, Solutions
```

```r
#=======================

#1)
out <- c() #Intitialize vector
for(i in 1:100){ #Loop from 1 to 100

    #Test if not multiple of 3 or 4
    if (i %% 3 != 0 && i %% 4 != 0){
        #Append to vector
        out <- c(out,i)
    }
}
#Print vector
out
```

```
##  [1]  1  2  5  7 10 11 13 14 17 19 22 23 25 26 29
## [16] 31 34 35 37 38 41 43 46 47 49 50 53 55 58 59
## [31] 61 62 65 67 70 71 73 74 77 79 82 83 85 86 89
## [46] 91 94 95 97 98
```

```r
#2)
fib_seq <- function(n){
    #This function takes integer n as input
    #And returns all Fibbonaci numbers less than n

    fibs <- c(1,1) #Initialize Fibbonaci sequence
    loop = TRUE
    while(loop==TRUE){
        #Calculate next fib number
        next_fib <- sum(tail(fibs,2))
        #If less than n, append to fib vector
        if(next_fib < n){
            fibs <- c(fibs,next_fib)
        #Else terminate loop
        } else {
        loop=FALSE
        }
    }
    #Return Fibbonaci sequence
    return(fibs)
}

fib_seq(1000) #Call function
```

```
##  [1]   1   1   2   3   5   8  13  21  34  55  89
## [12] 144 233 377 610 987
```

## 4 Basic Regression in R

Consider the linear model:

$$Y = X\beta + \epsilon$$

where $X$ is a scalar and $\epsilon$ is normally distributed. The code below can be used to simulate data from this model:

```
#=====================
#  TITLE:  computational economics: assignment 1

#  AUTHOR: John Eric Humphries

#  abstract: problem set on regression for econ 21410

#  Date: 2014-03-14
#=====================

#========================
# Section 0: setup
#========================
#setwd("")
rm(list=ls())            # Clear the workspace
set.seed(21410)          # Set random seed

library(ggplot2)
library(stargazer)
library(xtable)

#======================
# Section 1: Generating Data
#======================
n      <- 200              # observations
X      <- rnorm(n,20,10)
eps    <- rnorm(n,0,4)
beta   <- 3.1
const  <- 2
Y   <- const +  X * beta + eps
```

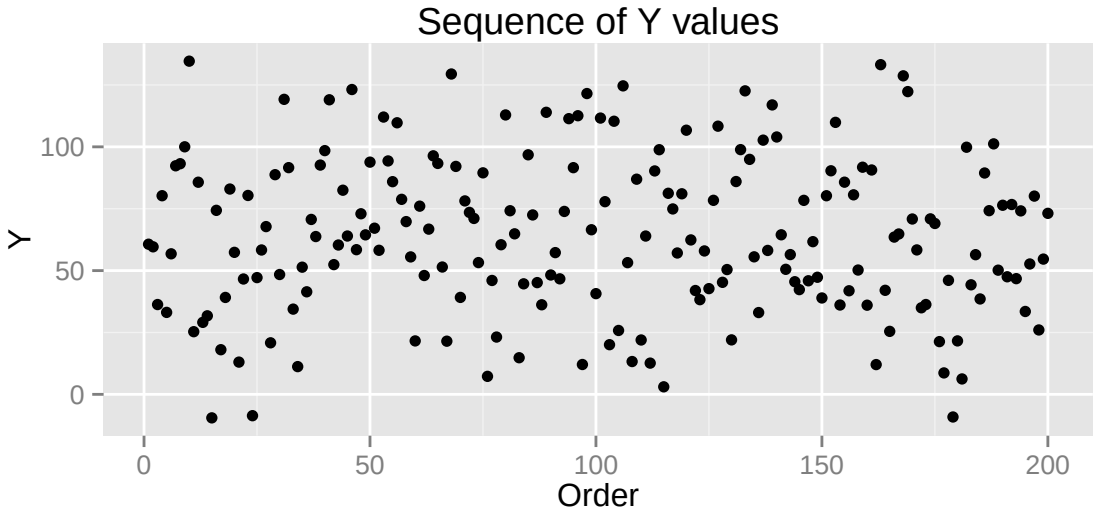
1. Calculate is the correlation between X and Y?
2. Plot the Y values for each individual (Y on the y-axis, 1-200 on the x-axis)
3. Plot a histogram of Y.
4. Plot a histogram of Y using the packages ggplot2 or ggvis.
5. Use your simulated data to run the regression of Y on X using the lm() command.
6. Make a latex table of the regression results using xtable() or stargazer()

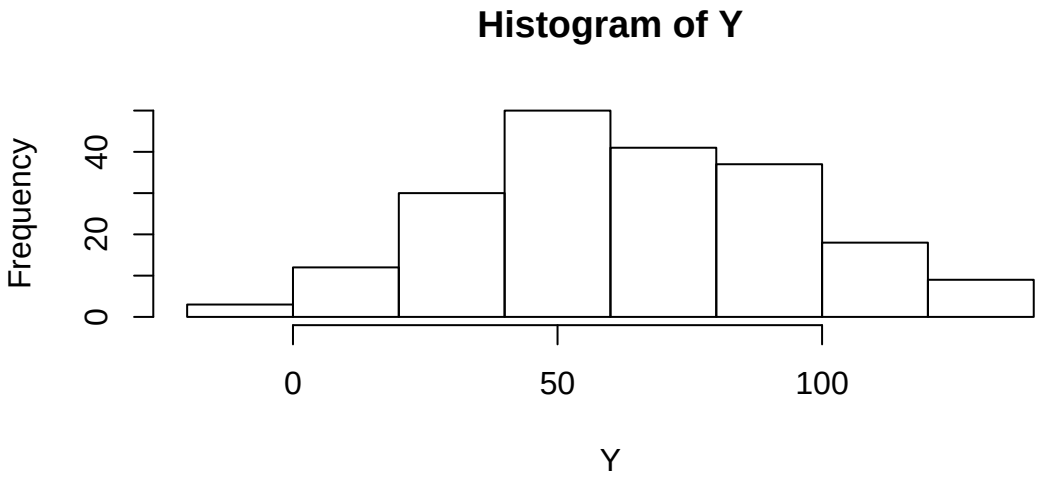
The Solution to 4:

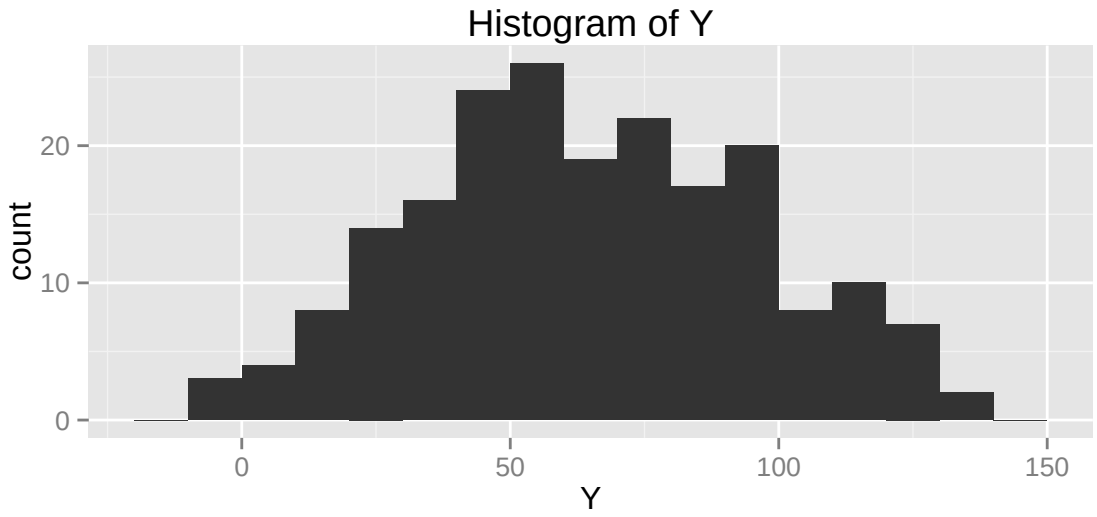
```
#========================
# Section 2: Solutions
#========================

#Correlation between X and Y
cor(X,Y)


## [1] 0.9909514


#Sequence of Values
qplot(1:200,Y)+xlab('Order')+ggtitle('Sequence of Y values')
#Histograms
hist(Y)
qplot(Y,geom="histogram",binwidth=10)+ggtitle('Histogram of Y')
#Scatter Plot
qplot(X,Y)+ggtitle('Scatter plot of X,Y')
#Define Regression
reg1 <- lm(Y~X)
#Regression Object
reg1


##
## Call:
## lm(formula = Y ~ X)
##
## Coefficients:
## (Intercept)            X
##       1.707        3.100


#Regression Summary
summary(reg1)


##
## Call:
## lm(formula = Y ~ X)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.1312  -3.0390  -0.1494   2.8018  12.1654
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.70729    0.67192   2.541   0.0118
## X            3.10036    0.02984 103.888   <2e-16
##
## (Intercept) *
```

```
## X           ***
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.259 on 198 degrees of freedom
## Multiple R-squared:  0.982,Adjusted R-squared:  0.9819
## F-statistic: 1.079e+04 on 1 and 198 DF,  p-value: < 2.2e-16


#Example stargazer table
stargazer(reg1)


##
## % Table created by stargazer v.5.1 by Marek Hlavac, Harvard University. E-mail: hlavac at
## % Date and time: Thu, Apr 09, 2015 - 14:53:13
## \begin{table}[!htbp] \centering
##   \caption{}
##   \label{}
## \begin{tabular}{@{\extracolsep{5pt}}lc}
## \\[-1.8ex]\hline
## \hline \\[-1.8ex]
##  & \multicolumn{1}{c}{\textit{Dependent variable:}} \\
## \cline{2-2}
## \\[-1.8ex] & Y \\
## \hline \\[-1.8ex]
##  X & 3.100$^{***}$ \\
##   & (0.030) \\
##   & \\
##  Constant & 1.707$^{**}$ \\
##   & (0.672) \\
##   & \\
## \hline \\[-1.8ex]
## Observations & 200 \\
## R$^{2}$ & 0.982 \\
## Adjusted R$^{2}$ & 0.982 \\
## Residual Std. Error & 4.259 (df = 198) \\
## F Statistic & 10,792.630$^{***}$ (df = 1; 198) \\
## \hline
## \hline \\[-1.8ex]
## \textit{Note:}  & \multicolumn{1}{r}{$^{*}$p$<$0.1; $^{**}$p$<$0.05; $^{***}$p$<$0.01} \\
## \end{tabular}
## \end{table}


#Example xtable table
xtable(reg1)


## % latex table generated in R 3.1.1 by xtable 1.7-4 package
## % Thu Apr  9 14:53:13 2015
```

```
## \begin{table}[ht]
## \centering
## \begin{tabular}{rrrrr}
##   \hline
##  & Estimate & Std. Error & t value & Pr($>$$|$t$|$) \\
##   \hline
## (Intercept) & 1.7073 & 0.6719 & 2.54 & 0.0118 \\
##   X & 3.1004 & 0.0298 & 103.89 & 0.0000 \\
##    \hline
## \end{tabular}
## \end{table}
```




Histogram of Y
count
20
10
0
0
50
100
150
Y

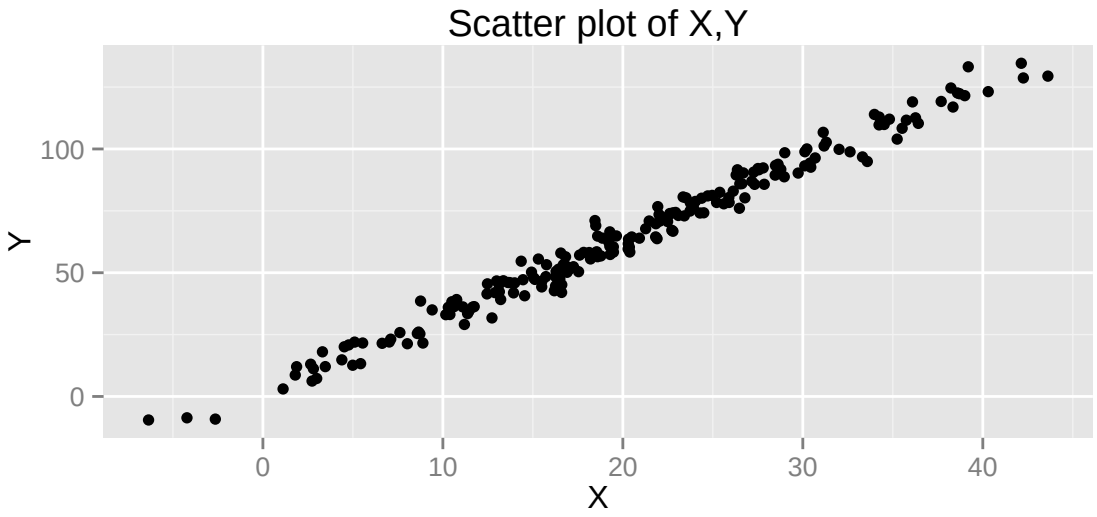

Scatter plot of X,Y
Y
100
50
0
0
10
20
30
40
X

Table 1: Stargazer Table

| | *Dependent variable:* |
|---|---|
| | Y |
| X | 3.100*** |
| | (0.030) |
| Constant | 1.707** |
| | (0.672) |
| Observations | 200 |
| $R^2$ | 0.982 |
| Adjusted $R^2$ | 0.982 |
| Residual Std. Error | 4.259 (df = 198) |
| F Statistic | 10,792.630*** (df = 1; 198) |
| *Note:* | *p<0.1; **p<0.05; ***p<0.01 |

Table 2: xtable Table

| | Estimate | Std. Error | t value | Pr(>\|t\|) |
|---|---|---|---|---|
| (Intercept) | 1.7073 | 0.6719 | 2.54 | 0.0118 |
| X | 3.1004 | 0.0298 | 103.89 | 0.0000 |

## 5 Getting Started with LaTeX

1. Insert an image off the internet into your latex file (preferably a kitten)

2. Display the matrix and vector $x$ and $y$ above in LaTeX(no need to include the decimals)

$$y = \begin{bmatrix} 200 \\ 100 \\ 500 \\ 400 \\ 300 \end{bmatrix}$$

$$x = \begin{bmatrix} 96 & 101 & 98 & 99 & 100 \\ 100 & 98 & 96 & 98 & 98 \\ 0 & 0 & 0 & 0 & 0 \\ 92 & 98 & 100 & 97 & 101 \\ 101 & 97 & 103 & 107 & 97 \end{bmatrix}$$

3. Print the symbols $\alpha$, $\theta_j$, $\lambda_{t,t+1}$, $\gamma^{s,t}$ inline with text.

4. Write on its own centered line:

$$\sum_{t=1}^{T} \frac{a_t}{b_t} \xrightarrow{p} \infty$$

5. Write $a \neq b$ and $c \geq d$

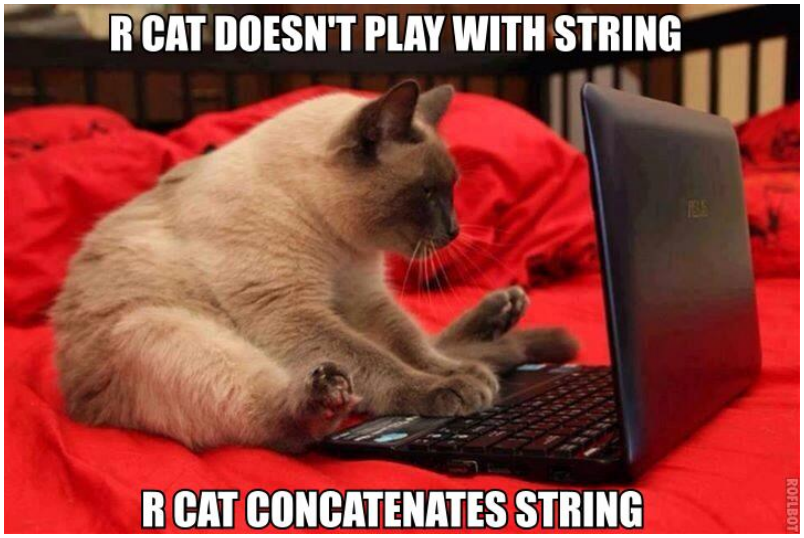



Figure 1: R cat

The Solution to 5:

```
\section{A Quick Review of \LaTeX }

\begin{enumerate}
    \item Insert an image off the internet into your latex file (preferably a kitten)
    \begin{figure}
        \centering
        \includegraphics[width = 0.5\paperwidth]{rkitty.jpg}
        \caption{R cat}
    \end{figure}

    \item Display the matrix and vector $x$ and $y$ above in \LaTeX
    (no need to include the decimals)
    \begin{eqnarray*}
        y & = & \left[\begin{array}{c}
        200\\
        100\\
        500\\
        400\\
        300
        \end{array}\right]\\
        x & = & \left[\begin{array}{ccccc}
        96  & 101 & 98  & 99  & 100\\
        100 & 98  & 96  & 98  & 98\\
        0   & 0   & 0   & 0   & 0\\
        92  & 98  & 100 & 97  & 101\\
        101 & 97  & 103 & 107 & 97
        \end{array}\right]
    \end{eqnarray*}

    \item Print the symbols $\alpha$, $\theta_j$, $\lambda_{t,t+1}$,
    $\gamma^{s,t}$ inline with text.
```

```
    \item Write on its own centered line:
    $$\sum_{t = 1}^T \frac{a_t}{b_t} \overset{p}{\rightarrow} \infty  $$

    \item Write $a \neq b$ and $c \geq d$

\end{enumerate}
```

## Becker's Marriage Market Warm Up

- 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 (all of this is assuming both first derivatives have the same sign, if they do not then the opposite converse is true).
- Non-Transferable Utility: In the non transferable case we assume that the two partners split the total utility by some pre-specified shares $\alpha$ and $1 - \alpha$. Now the person each individual wants to marry will depend on the sign of the partial derivatives. If both partial derivatives have the same sign, then we will get positive assortative mating. If the partial derivatives have the opposite signs we will get negative assoratative mating.

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 transferable utility case and (2) the non-transferable utility case. Explain how you reached your answers.
    - $h(m, f) = m^{0.3} f^{0.3}$
        - 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

## Simulating Becker's Marriage Market

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).
   - 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.
   - 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 $f$s than $m$s and that $m$'s "propose" in this model.[1] 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 (assume the output of each match is given by $h(m, f) = mf$) 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 $f$s get when education has the discrete binomial distribution (simulation 1)? 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 (simulation 2). How does this change the proportion of the output that the $f$s 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.

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 $f$s now get when education is binomially distributed (simulation 3)? How about when education is distributed uniformly (simulation 4)? How

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

does this differ from your result when there were fewer men than women (run the code to generate the data and your code a couple of times to make sure your result is not an outlier).
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.

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

# Create Matrix Structure for Output
set.seed(907)
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
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
```

```r
#  Uniform Distribution, More Males than Feales
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)
}

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))
}

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

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

    #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(nMales

    #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(nMales

    #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(nMales
}

#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, #Femal
colnames(outtable) <- c('Average Female share of Output','std.dev')
#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 men and women that consists of their rank in the matrix plus some random noise. To clarify, rankMale is an ordered list of which females each male prefers. The first row contains the ranking for male 1. For him,the first column is the index number of the female he prefers most, the second column is the index number for the women he prefers the second most, etc. For example, if the first three columns of the first row were 8,3,1, it would mean that the first male prefers the 8th female the most, followed by the 3rd, then the 1st, etc.The prior version of the code hand rankings listed by columns rather than rows, but this not how I wrote up rankings in class so I have modifiend the code below to match the description above.

```
# Section 4: Generate Agents with non-transferable utility and idiosyncratic component
#              for Gale-Syapley algorithm
#========================

rm(list=ls())
set.seed(907)
#Number of Agents
nMales      <- 20
nFemales    <- 30
#Match utility of agents (each column represents the utility of the agent
# when matched with the agent in the corresponding row)
utilMale    <- t(replicate(nMales,seq(100,1,length =nFemales)+100*runif(nFemales)))
utilFemale  <- t(replicate(nFemales,seq(100,1,length =nMales)+100*runif(nMales)))
#Match preference order of agents
rankMale    <- t(sapply(1:nMales,function(x) order(utilMale[x,],decreasing=T)))
rankFemale  <- t(sapply(1:nFemales,function(x) order(utilFemale[x,],decreasing=T)))

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

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

    - 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 a function that takes as inputs a matrix of men's rankings of women and a matrix of women's ranking of men as inputs and runs the Gale-Shapley algorithm to find the men-proposing stable match.

3. 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 for both genders.

4. Calcuate the total utility of men and women under this match.

5. Now add an argument *femaleProposal* to your function which when TRUE instead runs the female proposing Gale-Shapley algorithm. In which case do women have higher welfare? Why do you think this is?
Over all possible stable matches, in expectation men have the highest average welfare when they propose, and the lowest average welfare when the women propose (and vice versa). This can be theoretically proven, however our results are very noisy so we need to average over a large number of simulations to actually show this.

Table 3: Average welfare over 500 simulations

| | Male Utility | Female Utility |
|---|---|---|
| Men Propose | 6350 | 6080 |
| Female Propose | 6345 | 6086 |

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

DeferredAcceptanceAlgorithm <- function(males, females, females_propose = FALSE){
    # 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 other type
    #
    # 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

    if(females_propose){
        nProposers <- nrow(females)
        proposers  <- females
        nAcceptors <- nrow(males)
        acceptors  <- males
    } else {
        nProposers <- nrow(males)
        proposers  <- males
        nAcceptors <- nrow(females)
        acceptors  <- females
    }

    matches = matrix(0,nProposers,nAcceptors)
    prev_matches = matrix(1,nProposers,nAcceptors)
```

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

    if(females_propose){
        matches <- t(matches)
    }

    return(matches) # Return matches
}

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

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

#Number of Agents
nMales      <- 20
nFemales    <- 30

num.sim <- 500 #Number of Simulations

#Matrix for outputting the female share of each simulation
Util_sims <- matrix(rep(NA,4*num.sim),ncol=4)
colnames(Util_sims) <- c('maleUtil_maleProp','maleUtil_femaleProp'
                         ,'femaleUtil_maleProp','femaleUtil_femaleProp')
```

```r
for(i in 1:num.sim){

    #Randomly generate match utilities and ranks
    utilMale    <- t(replicate(nMales,seq(100,1,length =nFemales)+500*runif(nFemales)))
    utilFemale  <- t(replicate(nFemales,seq(100,1,length =nMales)+500*runif(nMales)))
    rankMale    <- t(sapply(1:nMales,function(x) order(utilMale[x,],decreasing=T)))
    rankFemale  <- t(sapply(1:nFemales,function(x) order(utilFemale[x,],decreasing=T)))

    #Male Proposal Matches
    malePropMatches <- DeferredAcceptanceAlgorithm(rankMale,rankFemale)
    #utility under Male Proposal Matches
    maleUtil_malePropose <- sum(utilMale*malePropMatches)
    femaleUtil_malePropose <- sum(t(utilFemale)*malePropMatches)

    #Female Proposal Matches
    femalePropMatches <- DeferredAcceptanceAlgorithm(rankMale,rankFemale,
                                                     females_propose=TRUE)
    #utility under Female Proposal Matches
    maleUtil_femalePropose <- sum(utilMale*femalePropMatches)
    femaleUtil_femalePropose <- sum(t(utilFemale)*femalePropMatches)

    #Store utilities in matrix
    Util_sims[i,] <- c(maleUtil_malePropose,maleUtil_femalePropose,
                       femaleUtil_malePropose,femaleUtil_femalePropose)
}
#Calculate average utilities

dat <- matrix(colMeans(Util_sims),nrow=2)
rownames(dat) <- c('Men Propose','Female Propose')
colnames(dat) <- c('Male Utility','Female Utility')
xtable(dat,digits=0)
```

## Potential Side Projects

Below are a list of potential side projects. On each homework you should include a "Side Projects" section at the end stating any projects you have completed over the last week. Additional files related to side projects should be additionally emaild to Oliver and myself.

- Complete your problem set in knitr (can only be done first week!). (0.5 points)
- Make a meaningful contribution to the class wiki, start an issue and ask a valuable question, provide a detailed and useful answer to a classmate's question. Include 1-3 sentences in your homework stating your contributions (can only be done first week!). (1 point)
- Rewrite at least a portion of pset code above in Julia[2] or C++ using the Rcpp package. Compare how long the new code takes to run in comparison with your R code. (3 points)

[2]Julia is a very promising new programming language for statistical computing. It is still very new, but I believe it may eventually be a quality replacement for R or python and some early investment now could be beneficial later. It is fast, has a simple syntax, is open source, and has a large community for such a young language.

- Rewrite a portion of pset code in python. (1.5 point)
- item Read Prof Becker's original 1974 paper on this subject and write a short (1-3 page) summary and response (up to 2.5 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 half to two page description of the idea (up to 2.5 points).