Thesis

Dadmehr Didgar

Preface

This document contains code snippets used in my thesis for generating various tables and graphs across different chapters. The code below is only a part of the comprehensive computational work carried out. Here, I'll focus on explaining key parts of the code.

Note on Package Loading with pacman::p_load()

The `pacman::p_load()` function is responsible for loading multiple R packages in one go. If a specified package is not installed, the function will install it before loading. This line is essentially setting up the environment by ensuring that all the required packages for the project are loaded and ready for use. The list of packages covers a wide range of functionalities including data manipulation, plotting, statistical analysis, and more.

```
# Load all necessary packages for analysis using the pacman package manager
pacman::p_load(ggplot2, ipumsr, dplyr, MASS, plotly, ggeasy, stargazer, usethis, tictoc, pryr, smoothie
```

Note on Data Preprocessing and Analysis Workflow

This script primarily focuses on loading and manipulating a dataset related to U.S. demographics, presumably obtained from IPUMS. Below is a breakdown of what the script does:

1. **Setting Working Directory**: The working directory is set to a specific folder on the user's computer.
2. **Package Requirement**: Checks for the `ipumsr` package and stops execution if the package is not found.
3. **Data Reading**: Reads metadata (`usa_00036.xml`) and actual microdata (`main_data`) using the IPUMS DDI (Data Documentation Initiative) schema.
4. **Data Subsetting**: Creates multiple subsets of the main data based on certain conditions, such as marital status, income, year, etc.
5. **Variable Creation**: Adds new variables (`mom`, `dad`, `parent`) to `main_data`.
6. **Quantile Analysis**: Creates income, age, race, and education grids by computing quantiles for multiple subsets (`m_main_data`, `jm_main_data`, `co_main_data`).
7. **Data Finalization**: Adjusts the generated grids by removing the first element.
8. **Year Selection**: Optionally filters the data by year (`n_year`).

Note: The use of `<<-` suggests that the variables are being created in the global environment, which may impact other parts of the code or R session.

```
setwd("/Users/dadmehr/R")
if (!require("ipumsr")) stop("Reading IPUMS data into R requires the ipumsr package. It can be installe
ddi <- read_ipums_ddi("usa_00036.xml")
main_data <- read_ipums_micro(ddi)
```

```
## Use of data from IPUMS USA is subject to conditions including that users should
## cite the data appropriately. Use command `ipums_conditions()` for more details.
```

```
m_main_data <- subset(main_data, INCTOT !=9999999 & MARST %in% c(1,2) & MARRINYR != 2 & YEAR==2021)
jm_main_data <- subset(main_data, INCTOT !=9999999 & MARRINYR == 2 & YEAR==2021)
co_main_data <- subset(main_data, INCTOT !=9999999 & COUPLETYPE == 3 & MARST == 6 & YEAR==2021)
co_2021 <- subset(main_data, INCTOT !=9999999 & YEAR==2021 & COUPLETYPE == 3 & MARST == 6)

main_data <- main_data %>% mutate(mom = case_when(MOMLOC > 0 ~ 1, !MOMLOC > 0 ~ 0))
main_data <- main_data %>% mutate(dad = case_when(POPLOC > 0 ~ 1, !POPLOC > 0 ~ 0))
main_data <- main_data %>% mutate(parent = case_when(abs(dad+mom) > 0 ~ 1, !abs(dad+mom) > 0 ~ 0))



cut_by <- 0.1

m_income_grid <<- sort(unique(as.integer(c(quantile(m_main_data$INCTOT, probs = seq(0, 1, by = cut_by))
m_age_grid <<- sort(unique(as.integer(c(quantile(m_main_data$AGE, probs = seq(0, 1, by = cut_by))))))
m_race_grid <<- sort(unique(as.integer(c(quantile(m_main_data$RACE, probs = seq(0, 1, by = 0.2))))))
m_edu_grid <<- sort(unique(as.integer(c(quantile(m_main_data$EDUCD, probs = seq(0, 1, by = 0.2))))))

m_income_grid <- m_income_grid[-1]
m_age_grid <- m_age_grid[-1]
m_race_grid <- m_race_grid[-1]
m_edu_grid <- m_edu_grid[-1]

jm_income_grid <<- sort(unique(as.integer(c(quantile(jm_main_data$INCTOT, probs = seq(0, 1, by = cut_by
jm_age_grid <<- sort(unique(as.integer(c(quantile(jm_main_data$AGE, probs = seq(0, 1, by = cut_by))))))
jm_race_grid <<- sort(unique(as.integer(c(quantile(jm_main_data$RACE, probs = seq(0, 1, by = 0.2))))))
jm_edu_grid <<- sort(unique(as.integer(c(quantile(jm_main_data$EDUCD, probs = seq(0, 1, by = 0.2))))))

jm_income_grid <- jm_income_grid[-1]
jm_age_grid <- jm_age_grid[-1]
jm_race_grid <- jm_race_grid[-1]
jm_edu_grid <- jm_edu_grid[-1]

co_income_grid <<- sort(unique(as.integer(c(quantile(co_main_data$INCTOT, probs = seq(0, 1, by = cut_by
co_age_grid <<- sort(unique(as.integer(c(quantile(co_main_data$AGE, probs = seq(0, 1, by = cut_by))))))
co_race_grid <<- sort(unique(as.integer(c(quantile(co_main_data$RACE, probs = seq(0, 1, by = 0.2))))))
co_edu_grid <<- sort(unique(as.integer(c(quantile(co_main_data$EDUCD, probs = seq(0, 1, by = 0.2))))))

co_income_grid <- co_income_grid[-1]
co_age_grid <- co_age_grid[-1]
co_race_grid <- co_race_grid[-1]
co_edu_grid <- co_edu_grid[-1]


m_race_grid <<- jm_race_grid <<- co_race_grid <<- race_gridrace_grid <<- c(1,2,6,9)
m_edu_grid <<- jm_edu_grid <<- co_edu_grid <<- edu_grid <<- c(63,81,101,116)

n_age <<- length(jm_age_grid)
n_income <<- length(jm_income_grid)
n_edu <<- length(jm_edu_grid)
n_race <<- length(jm_race_grid)

n_year=2021
```

```
if (n_year!=-1){data <- subset(main_data,main_data$YEAR == n_year)}
```

Note on Functions and Label Mappings

This script block contains two primary components:

1. **Gumbel Functions**: Three functions (`F_gumbel`, `gumbel`, and `G_gumbel`) are defined for Gumbel distribution calculations.
   - `F_gumbel(n)`: Calculates the Gumbel distribution's cumulative distribution function (CDF) using input `n`.
   - `gumbel(n)`: Essentially identical to `F_gumbel`, calculates the Gumbel CDF.
   - `G_gumbel(n)`: Calculates another variant of the Gumbel distribution, making use of the `gumbel` function.
2. **Label Mappings**: Two vectors (`race_labels` and `edu_labels`) are created for mapping numerical codes to human-readable category labels.
   - `race_labels`: Maps numerical codes to racial categories.
   - `edu_labels`: Maps numerical codes to various educational levels.

These functions and label mappings are likely used later in the code to facilitate statistical analysis and data visualization.

```
F_gumbel <- function(n){
  return(exp(-exp(-n)))
}

gumbel <- function(n){
  return(exp(-exp(-n)))
}

G_gumbel <- function(n){
  a <- gumbel(n)+(1-gumbel(n))/2
  return(-(log(log(1/a))))
}

race_labels = c("1" = "White","2" = "Black",
                "3" = "Indian/Native","4" = "Chinese",
                "5" = "Japanese","6" = "Other Asian",
                "7" = "Other","8" = "Two races","9" = "Three or more")

edu_labels = c('0' = 'No schooling',
               '1' = 'N/A',
               '2' = 'No schooling',
               '10' = 'Nursery to grade 4',
               '11' = 'Nursery',
               '12' = 'Kindergarten',
               '13' = 'Grade 1-4',
               '14' = 'Grade 1',
               '15' = 'Grade 2',
               '16' = 'Grade 3',
               '17' = 'Grade 4',
               '20' = 'Grade 5-8',
               '21' = 'Grade 5-6',
               '22' = 'Grade 5',
               '23' = 'Grade 6',
               '24' = 'Grade 7-8',
```

```
                '25' = 'Grade 7',
                '26' = 'Grade 8',
                '30' = 'Grade 9',
                '40' = 'Grade 10',
                '50' = 'Grade 11',
                '60' = 'Grade 12',
                '61' = '12th grade, no diploma',
                '62' = 'High school graduate or GED',
                '63' = 'High school diploma',
                '64' = 'GED',
                '65' = 'Some college',
                '70' = '1 year of college',
                '71' = 'College credit',
                '80' = '2 years of college',
                '81' = 'Associate',
                '82' = 'Associate, occupational',
                '83' = 'Associate, academic',
                '90' = '3 years of college',
                '100' = '4 years of college',
                '101' = 'Bachelor',
                '110' = '5+ years of college',
                '111' = '6 years of college',
                '112' = '7 years of college',
                '113' = '8+ years of college',
                '114' = 'Master',
                '115' = 'Professional',
                '116' = 'Doctoral')
```

Note on Setting Parameters and Identifying Major Cities

This code block focuses on two main tasks:

1. **Parameter Initialization**: Several global variables are initialized to set various parameters for later analysis or calculations. For example, `precision` is set to 0.01, and `income_max` is set to 1,265,000. These variables serve as configuration settings that are likely used throughout the data analysis pipeline.
   - `precision`, `small_epsilon`: Precision settings for calculations.
   - `age_distance`, `income_distance`: Define the acceptable distance or gap in age and income for comparisons.
   - `income_max`, `income_min`: Define the income range.
   - `age_max`, `age_min`, `big_age`: Define the age range.
2. **Major City Codes**: The `major_city` vector is populated with numerical codes that correspond to major U.S. cities. Each number represents a specific city (e.g., 4610 for New York, NY, 3730 for Los Angeles, CA, etc.).

The variables and city codes set up in this block serve as foundational settings for later data manipulation and analysis.

```
precision <<- 0.01
small_epsilon <<- 0.9
age_distance <<- 2
income_distance <<- 10000
income_max <<- 1265000
income_min <<- 0
age_max <<- 95
age_min <<- 18
```

```r
big_age <<- 19

major_city <<- c(4610,3730,1190,5350,5330,6290,2990,6430,3110)

#
# 1    0 [Not in identifiable city (or size group)] 2322246
# 2 4610 [New York, NY]                                 58688
# 3 3730 [Los Angeles, CA]                              31761
# 4 1190 [Chicago, IL]                                  19216
# 5 5350 [Phoenix, AZ]                                  10434
# 6 5330 [Philadelphia, PA]                              8794
# 7 6290 [San Francisco, CA]                             6590
# 8 2990 [Indianapolis, IN]                              6030
# 9 6430 [Seattle, WA]                                   5801
# 10 3110 [Jacksonville, FL]                              5680
# 11  810 [Boston, MA]                                    5416
# 12 1710 [Denver, CO]                                    5411
# 13 7230 [Washington, DC]                                5343
# 14 5530 [Portland, OR]                                  5289
# 15 4410 [Nashville-Davidson, TN]                        4870
# 16 4050 [Mesa, AZ]                                      4455
# 17  530 [Baltimore, MD]                                 4297
# 18 2010 [El Paso, TX]                                   4088
# 19 1750 [Detroit, MI]                                   4003
# 20 4930 [Oakland, CA]                                   3978
```

Demand Estimation in the Marriage Market Using Logit Model

The following code is the crux of the entire thesis. It aims to estimate demand in the marriage market using a Discrete Choice Model based on the Logit function. This demand estimation serves as a crucial step to understand the dynamics of the marriage market, including the likelihood of specific pairings given the attributes of individuals within the market.

Key Components:

- **Data Preparation**: The code assumes that data for individuals in the marriage market, represented as `df`, is already cleaned and preprocessed. Each row represents an individual with attributes like `age`, `education`, `income`, etc.
- **Griding Parameters**: In order to estimate the demand effectively, a grid search is employed to find the optimal parameters (`alpha`, `beta_age`, `beta_education`, etc.) that maximize the likelihood function. These parameters represent the weight of each attribute in determining a match.
- **Logit Function**: The core mathematical model used is the Logit model, which transforms the linear combination of variables into a probability outcome. This is implemented in the function `logit_demand_estimation`.
- **Maximum Likelihood Estimation (MLE)**: MLE is used to find the parameters that maximize the likelihood of observing the sample data. The `optimize` function is used for this purpose.
- **Output**: The function returns the optimal parameters, which can be interpreted to understand the factors that significantly influence demand in the marriage market.

```r
funk <- function(n_year){
```

```r
income_grid <<- jm_income_grid
age_grid <<- jm_age_grid
edu_grid <<- jm_edu_grid
race_grid <<- jm_race_grid

n_age <<- length(age_grid)
n_income <<- length(income_grid)
n_edu <<- length(edu_grid)
n_race <<- length(race_grid)


if (n_year!=2021){data <- subset(main_data,main_data$YEAR == n_year)}
data$income <- data$INCTOT/1000



if (n_year == 2021){
  states_codes <- as.integer(sort(unique(data$STATEICP)))
  n_states <- length(states_codes)
  last_state_code <- states_codes[n_states]
  states_gini <- rep(0,last_state_code)
  states_singles_gini <- rep(0,last_state_code)
  for (i in c(1:n_states)){
    state_code <- states_codes [i]
    #print(i)
    state_data <- subset(data, data$STATEICP == state_code )
    states_gini[state_code] <- gini.wtd(state_data$INCTOT, weights = NULL)
    state_single_data <- subset(data, data$STATEICP == state_code &
                              (data$MARST == 6  | (data$MARST %in% c(1,2) & data$MARRINYR == 2))
    )
    states_singles_gini[state_code] <- gini.wtd(state_single_data$INCTOT, weights = NULL)
  }

  data$STATEGINI <- states_gini[data$STATEICP]
  data$SINGLEGINI <- states_singles_gini[data$STATEICP]



  city_codes <- as.integer(sort(unique(data$CITY)))
  n_city <- length(city_codes)
  last_city_code <- city_codes[n_city]
  city_gini <- rep(0,last_city_code)
  city_single_gini <- rep(0,last_city_code)
  for (i in c(1:n_city)){
    city_code <- city_codes [i]
    #print(i)
    city_data <- subset(data, data$CITY == city_code )
    city_gini[city_code] <- gini.wtd(city_data$INCTOT, weights = NULL)
    city_single_data <- subset(data, data$CITY == city_code &
                              (data$MARST == 6  | (data$MARST %in% c(1,2) & data$MARRINYR == 2))
    )
    city_single_gini[city_code] <- gini.wtd(city_single_data$INCTOT, weights = NULL)
  }
```

```r
  data$CITYGINI <- 0
  data$CITYGINI[data$CITY>0] <- city_gini[data$CITY]
  data$CITYSINGLEGINI <- 0
  data$CITYSINGLEGINI[data$CITY>0] <- city_single_gini[data$CITY]

}

pool_data <- subset(data, (data$MARST == 6 &  data$AGE > 15 )
                    | data$RELATE %in% c(1,2)
                    | (data$MARST %in% c(1,2)
                      & data$MARRINYR == 2
                      & data$AGE > 15)
)
pool_data <- pool_data %>% mutate(out = case_when(MARRINYR ==2 ~ 1,
                                                  MARRINYR !=2 ~ 0))



nm_data <- subset(data,data$MARST == 6 &  data$AGE > 15 )
nmf_data <- subset(data,data$MARST == 6  & data$SEX == 2 & data$AGE > 15 )
nmm_data <- subset(data,data$MARST == 6  & data$SEX == 1 & data$AGE > 15 )
jm_data <- subset(data,data$MARST %in% c(1,2)
                  & data$MARRINYR == 2 # make it == for jm
                  & data$RELATE %in% c(1,2) # new
                  & data$AGE > 15 )
jmf_data <- subset(data,data$MARST %in% c(1,2)
                   & data$MARRINYR == 2  # make it == for jm
                   & data$RELATE %in% c(1,2) # new
                   & data$SEX == 2 & data$AGE > 15)
jmm_data <- subset(data,data$MARST %in% c(1,2)
                   & data$MARRINYR == 2
                   & data$RELATE %in% c(1,2) # new
                   & data$SEX == 1 & data$AGE > 15)
jm_pair_data <- merge(jmf_data,jmm_data,by=c("SERIAL"),all=FALSE)

all_jm_pair_data <- merge(jm_data,jm_data,by=c("SERIAL"),all=FALSE)
dim(all_jm_pair_data)

all_jm_pair_data <- subset(all_jm_pair_data,all_jm_pair_data$SEX.x == all_jm_pair_data$SEX.y)
dim(all_jm_pair_data)

all_jm_pair_data <- subset(all_jm_pair_data,all_jm_pair_data$NUMPREC.x != all_jm_pair_data$NUMPREC.y)
dim(all_jm_pair_data)


pair_data <- jm_pair_data
#dim(pair_data) # 12202
single_data <- nm_data


jm_data$NEW_INCTOT <- jm_data$INCTOT
jm_data$NEW_INCTOT <- cut(jm_data$NEW_INCTOT,
                          breaks=c(-Inf,income_grid),
```

```
                            labels=seq(1:n_income))
                            #breaks=c(income_grid),
                            #labels=seq(1:n_income-1))

# XXX
pair_data$NEW_AGE.x <- pair_data$AGE.x
pair_data$NEW_AGE.x <- cut(pair_data$NEW_AGE.x,
                           breaks=c(-Inf,age_grid),
                           labels=seq(1:n_age))
                           #breaks=c(age_grid),
                           #labels=seq(1:n_age-1))


pair_data$NEW_INCTOT.x <- pair_data$INCTOT.x
pair_data$NEW_INCTOT.x <- cut(pair_data$NEW_INCTOT.x,
                              breaks=c(-Inf,income_grid),
                              labels=seq(1:n_income))
                              #breaks=c(income_grid),
                              #labels=seq(1:n_income-1))

pair_data$NEW_EDUCD.x <- pair_data$EDUCD.x
pair_data$NEW_EDUCD.x <- cut(pair_data$NEW_EDUCD.x,
                             breaks=c(-Inf, edu_grid),
                             labels=seq(1:n_edu))

pair_data$NEW_RACE.x <- pair_data$RACE.x
pair_data$NEW_RACE.x <- cut(pair_data$NEW_RACE.x,
                            breaks=c(-Inf, race_grid),
                            labels=seq(1:n_race))

#YYY
pair_data$NEW_AGE.y <- pair_data$AGE.y
pair_data$NEW_AGE.y <- cut(pair_data$NEW_AGE.y,
                           breaks=c(-Inf,age_grid),
                           labels=seq(1:n_age))

pair_data$NEW_INCTOT.y <- pair_data$INCTOT.y
pair_data$NEW_INCTOT.y <- cut(pair_data$NEW_INCTOT.y,
                              breaks=c(-Inf,income_grid),
                              labels=seq(1:n_income))

pair_data$NEW_EDUCD.y <- pair_data$EDUCD.y
pair_data$NEW_EDUCD.y <- cut(pair_data$NEW_EDUCD.y,
                             breaks=c(-Inf, edu_grid),
                             labels=seq(1:n_edu))

pair_data$NEW_RACE.y <- pair_data$RACE.y
pair_data$NEW_RACE.y <- cut(pair_data$NEW_RACE.y,
                            breaks=c(-Inf, race_grid),
                            labels=seq(1:n_race))

Mu_pair <- count(pair_data,
                 NEW_AGE.x,NEW_INCTOT.x,NEW_EDUCD.x,NEW_RACE.x,
```

```r
                    NEW_AGE.y,NEW_INCTOT.y,NEW_EDUCD.y,NEW_RACE.y)#,.drop = FALSE)


#Mu_pair

#       MALE
single_male <- subset(single_data,single_data$SEX == 1)

single_male$NEW_AGE <- single_male$AGE
single_male$NEW_AGE <- as.integer(cut(single_male$NEW_AGE,
                                      breaks=c(-Inf,age_grid),
                                      labels=seq(1:n_age)))

single_male$NEW_INCTOT <- single_male$INCTOT
single_male$NEW_INCTOT <- as.integer(cut(single_male$NEW_INCTOT,
                                         breaks=c(-Inf,income_grid),
                                         labels=seq(1:n_income)))

single_male$NEW_EDUCD <- single_male$EDUCD
single_male$NEW_EDUCD <- as.integer(cut(single_male$NEW_EDUCD,
                                        breaks=c(-Inf, edu_grid),
                                        labels=seq(1:n_edu)))

single_male$NEW_RACE <- single_male$RACE
single_male$NEW_RACE <- as.integer(cut(single_male$NEW_RACE,
                                       breaks=c(-Inf, race_grid),
                                       labels=seq(1:n_race)))

#single_male <- na.omit(single_male, c("NEW_INCTOT","NEW_EDUCD","NEW_RACE", "NEW_AGE"))

Mu_male <- count(single_male,NEW_AGE,NEW_INCTOT,NEW_EDUCD,NEW_RACE) #,.drop = FALSE)
#Mu_male=count(single_male,NEW_AGE,NEW_INCTOT,NEW_EDUCD,NEW_RACE,.drop = FALSE)

#Mu_male

colnames(Mu_male) <- c("NEW_AGE.y","NEW_INCTOT.y","NEW_EDUCD.y","NEW_RACE.y","n_male")

#        FEMALE
single_female <- subset(single_data,single_data$SEX == 2)

single_female$NEW_AGE <- single_female$AGE
single_female$NEW_AGE <- as.integer(cut(single_female$NEW_AGE,
                                        breaks=c(-Inf,age_grid),
                                        labels=seq(1:n_age)))

single_female$NEW_INCTOT <- single_female$INCTOT
single_female$NEW_INCTOT <- as.integer(cut(single_female$NEW_INCTOT,
                                           breaks=c(-Inf,income_grid),
                                           labels=seq(1:n_income)))

single_female$NEW_EDUCD <- single_female$EDUCD
single_female$NEW_EDUCD <- as.integer(cut(single_female$NEW_EDUCD,
                                          breaks=c(-Inf, edu_grid),
```

```
                                          labels=seq(1:n_edu)))

single_female$NEW_RACE <- single_female$RACE
single_female$NEW_RACE <- as.integer(cut(single_female$NEW_RACE,
                                         breaks=c(-Inf, race_grid),
                                         labels=seq(1:n_race)))


Mu_female <- count(single_female,NEW_AGE,NEW_INCTOT,NEW_EDUCD,NEW_RACE) #,.drop = FALSE

#Mu_female

colnames(Mu_female) <- c("NEW_AGE.x","NEW_INCTOT.x","NEW_EDUCD.x","NEW_RACE.x","n_female")
#Mu_female

Mu_all <- merge(Mu_pair,Mu_female,by=c("NEW_AGE.x","NEW_INCTOT.x","NEW_EDUCD.x","NEW_RACE.x"),all=TRU
Mu_all <- merge(Mu_all,Mu_male,by=c("NEW_AGE.y","NEW_INCTOT.y","NEW_EDUCD.y","NEW_RACE.y"),all=TRUE)
Mu_all <- subset(Mu_all,!is.na(Mu_all$n))

Mu_all$n_male[is.na(Mu_all$n_male)] <- small_epsilon

Mu_all$n_female[is.na(Mu_all$n_female)] <- small_epsilon

Mu_all$MV <- ((Mu_all$n/Mu_all$n_male)*(Mu_all$n/Mu_all$n_female))^0.5

Mu_all$EV <- Mu_all$MV*Mu_all$n
Mu_all$MV_male <- Mu_all$n/Mu_all$n_male
Mu_all$EV_male <- Mu_all$MV_male*Mu_all$n
Mu_all$MV_female <- Mu_all$n/Mu_all$n_female
Mu_all$EV_female <- Mu_all$MV_female*Mu_all$n

#log version
Mu_all$MV_log <- log(Mu_all$MV)
Mu_all$EV_log <- Mu_all$MV_log*Mu_all$n

#Expected
Mu_all$MV_log_expected <- Mu_all$MV_log + G_gumbel(-Mu_all$MV_log)
Mu_all$EV_log_expected <- Mu_all$MV_log_expected*Mu_all$n

Mu_all$MV_male_log <- log(Mu_all$MV_male)
Mu_all$EV_male_log  <- Mu_all$MV_male_log*Mu_all$n

#Expected
Mu_all$MV_male_log_expected <- Mu_all$MV_male_log + G_gumbel(-Mu_all$MV_male_log)
Mu_all$EV_male_log_expected <- Mu_all$MV_male_log_expected*Mu_all$n

Mu_all$MV_female_log <- log(Mu_all$MV_female)
Mu_all$EV_female_log  <- Mu_all$MV_female_log*Mu_all$n

#Expected
Mu_all$MV_female_log_expected <- Mu_all$MV_female_log + G_gumbel(-Mu_all$MV_female_log)
Mu_all$EV_female_log_expected <- Mu_all$MV_female_log_expected*Mu_all$n
```

```r
Mu_all$tau <- (Mu_all$MV_male-Mu_all$MV_female)/2

#this one correct?
Mu_all$tau <- (Mu_all$MV_male_log-Mu_all$MV_female_log)/2

#Expected
Mu_all$tau_expected <- (Mu_all$MV_male_log_expected - Mu_all$MV_female_log_expected)/2


theta_min <- as.numeric(quantile(Mu_all$tau,probs=0))
theta_max <- as.numeric(quantile(Mu_all$tau,probs=1))

Mu_all$theta <- (Mu_all$tau- theta_min)/(theta_max-theta_min)
Mu_all$theta[Mu_all$theta<0] <- 0
Mu_all$theta[Mu_all$theta>1] <- 1


Mu_all$female_net_gain <- Mu_all$MV_female_log+Mu_all$tau
Mu_all$male_net_gain <- Mu_all$MV_female_log-Mu_all$tau

Mu_all$male_share <- (Mu_all$MV_male_log-Mu_all$tau)/Mu_all$MV_log
Mu_all$female_share <- (Mu_all$MV_female_log+Mu_all$tau)/Mu_all$MV_log

col_names <- c("NEW_INCTOT.x", "NEW_INCTOT.y","NEW_AGE.x", "NEW_AGE.y",
               "NEW_RACE.x", "NEW_RACE.y","NEW_EDUCD.x", "NEW_EDUCD.y")

pair_data <- merge(pair_data, Mu_all, by.x=col_names,
                   by.y=col_names,all.x=TRUE)

pair_data <- pair_data %>%
  set_variable_labels(
    NEW_AGE.x = "Female Age",
    NEW_AGE.y = "Male Age",
    NEW_INCTOT.x = "Female Income",
    NEW_INCTOT.y = "Male Income",
    income.x = "Female Income",
    income.y = "Male Income",
    NEW_EDUCD.x = "Female Education",
    NEW_EDUCD.y = "Male Education",
    NEW_RACE.x = "Female Race",
    NEW_RACE.y = "Male Race"
  )

pair_data$age_dif <- pair_data$AGE.x-pair_data$AGE.y
pair_data$income_dif <- pair_data$INCTOT.x-pair_data$INCTOT.y
pair_data$edu_dif <- pair_data$EDUCD.x-pair_data$EDUCD.y


pair_data$new_age_dif <- as.integer(pair_data$NEW_AGE.x)-as.integer(pair_data$NEW_AGE.y)
pair_data$new_income_dif <- as.integer(pair_data$NEW_INCTOT.x)-as.integer(pair_data$NEW_INCTOT.y)
pair_data$new_edu_dif <- as.integer(pair_data$NEW_EDUCD.x)-as.integer(pair_data$NEW_EDUCD.y)

pair_data$big_age_gap <- 0
```

```
  pair_data$big_age_gap[abs(pair_data$age_dif)>big_age] <- 1


  pair_data$cut_age_dif <- cut(pair_data$age_dif,
                               breaks=c(0,10,20,30,40,50,60,70),
                               labels=c(0,1,2,3,4,5,6))



  pair_data$theta_prime <- pair_data$theta/(1-pair_data$theta+0.01)

  pair_data$c_m <- 1/(pair_data$dad.y+0.01)
  pair_data$c_f <- 1/(pair_data$dad.x+0.01)
  pair_data$c_t <- pair_data$c_m+pair_data$c_f
  pair_data$LHS <- pair_data$c_m - pair_data$theta_prime*pair_data$c_f
  pair_data$RHS <- - pair_data$theta_prime*pair_data$c_t




  pair_data$c_m_2 <- pair_data$TRANWORK.y
  pair_data$c_f_2 <- pair_data$TRANWORK.x
  pair_data$c_t_2 <- pair_data$c_m_2+pair_data$c_f_2
  pair_data$LHS_2 <- pair_data$c_m_2 - pair_data$theta_prime*pair_data$c_f_2
  pair_data$RHS_2 <- - pair_data$theta_prime*pair_data$c_t_2

  # Nash bargenning -> tau=(alpha-gamma)/2

  pair_data$rate <- F_gumbel(-pair_data$MV_log)
  pair_data$rate.x <- F_gumbel(-pair_data$MV_female)
  pair_data$rate.y <- F_gumbel(-pair_data$MV_male)

  pair_data <- pair_data %>% mutate(major_city.x = case_when(CITY.x %in% major_city ~ 1,
                                                            !CITY.x %in% major_city ~ 0))
  pair_data <- pair_data %>% mutate(major_city.y = case_when(CITY.y %in% major_city ~ 1,
                                                            !CITY.y %in% major_city ~ 0))


  pair_data <- pair_data %>% mutate(big_city.x = case_when(CITY.x > 0 ~ 1,
                                                          !CITY.x > 0 ~ 0))
  pair_data <- pair_data %>% mutate(big_city.y = case_when(CITY.y > 0 ~ 1,
                                                          !CITY.y >0  ~ 0))

  pair_data$f_income_share <- pair_data$income.x/(pair_data$income.x+pair_data$income.y)

  output <- list("output",
                 pair_data = pair_data,
                 Mu_all = Mu_all)

  return(output)
}
```

Analysis of the R Code for Data Visualization

Overview

This R code generates and visualizes data on various demographic grids (e.g. age, income, education) through 3D plots.

Main Components

Initialization

- The year is set to 2021 and global variables are established.

Function Definition: `draw`

Tasks performed by the `draw` function include:

Local Variables

- Age, income, education, and race grids are established.

Data Retrieval

- Data is fetched from the `funk` function and stored in local variables `Mu_all` and `pair_data`.

Specific Computations

Age-based Computations

1. Calculates mutual gain matrices based on age grids.
   - Each matrix cell stores the expected mutual gain for a specific combination of ages for males and females.
2. Smoothening matrices using a 2D kernel smoother.
3. Visualization of smoothened matrices using the `persp` function for 3D plots.

Income-based Computations

- Same steps as age-based computations, but with income grids.

Education-based Computations

- The code prepares similar calculations for education grids.

Visualizations

- The `persp` function is used extensively for 3D perspective plots.
  - **X and Y Axes:** Different groups (e.g., male and female age levels)
  - **Z Axis:** Some form of calculated mutual gain

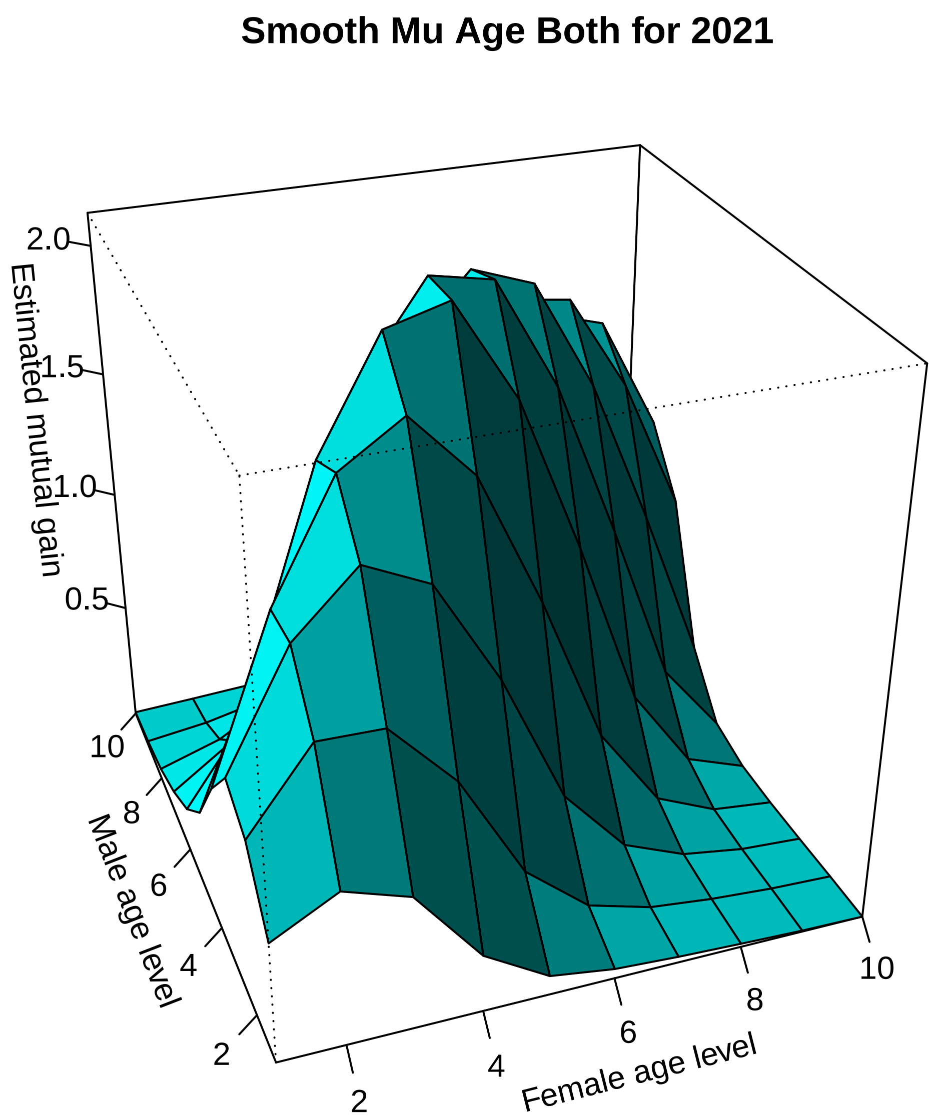
Conclusion

This script serves as an exploratory data analysis tool for visualizing mutual gains across different demographic grids.

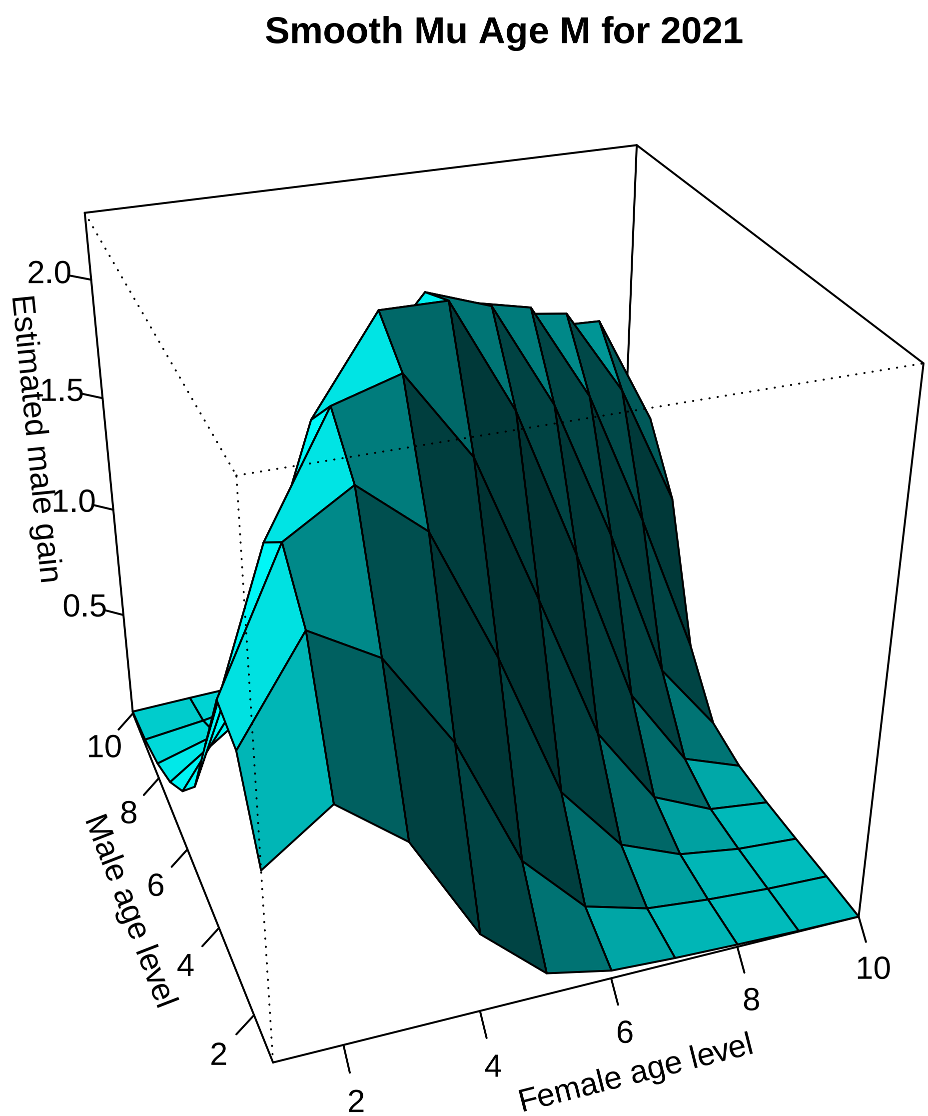
```
n_year <<- 2021




income_grid <<- jm_income_grid
age_grid <<- jm_age_grid
edu_grid <<- jm_edu_grid
race_grid <<- jm_race_grid

n_age <<- length(age_grid)
n_income <<- length(income_grid)
n_edu <<- length(edu_grid)
n_race <<- length(race_grid)



store <- funk(n_year)
Mu_all <- store$Mu_all
pair_data <- store$pair_data



####           AGE



N_Age <- matrix(0, n_age, n_age)
Mu_age <- matrix(0, n_age, n_age)
Mu_M_age <- matrix(0, n_age, n_age)
Mu_F_age <- matrix(0, n_age, n_age)
#log version
Mu_age_log <- matrix(0, n_age, n_age)
Mu_M_age_log <- matrix(0, n_age, n_age)
Mu_F_age_log <- matrix(0, n_age, n_age)

#ave
Mu_age_ave <- matrix(0, n_age, n_age)
Mu_M_age_ave <- matrix(0, n_age, n_age)
Mu_F_age_ave <- matrix(0, n_age, n_age)
#ave log version
Mu_age_log_ave <- matrix(0, n_age, n_age)
Mu_M_age_log_ave <- matrix(0, n_age, n_age)
Mu_F_age_log_ave <- matrix(0, n_age, n_age)

#median
Mu_age_median <- matrix(0, n_age, n_age)
Mu_M_age_median <- matrix(0, n_age, n_age)
Mu_F_age_median <- matrix(0, n_age, n_age)
#median log version
```

```r
Mu_age_log_median <- matrix(0, n_age, n_age)
Mu_M_age_log_median <- matrix(0, n_age, n_age)
Mu_F_age_log_median <- matrix(0, n_age, n_age)


for (i in 1:n_age){
  for (j in 1:n_age){
    #use pi
    current_sub <- subset(Mu_all, Mu_all$NEW_AGE.x == i & Mu_all$NEW_AGE.y == j)

    total_folks = 1 # sum(current_sub$n)

    Mu_age_log[i,j] = sum(current_sub$EV_log_expected) / total_folks
    Mu_F_age_log[i,j] = sum(current_sub$EV_female_log_expected) / total_folks
    Mu_M_age_log[i,j] = sum(current_sub$EV_male_log_expected) / total_folks

    Mu_age[i,j] = sum(current_sub$EV) / total_folks
    Mu_F_age[i,j] = sum(current_sub$EV_female) / total_folks
    Mu_M_age[i,j] = sum(current_sub$EV_male) / total_folks
    N_Age[i,j] = dim(current_sub)[1]


    # median

    Mu_age_log_median[i,j] = median(current_sub$EV_log_expected) / total_folks
    Mu_F_age_log_median[i,j] = median(current_sub$EV_female_log_expected) / total_folks
    Mu_M_age_log_median[i,j] = median(current_sub$EV_male_log_expected) / total_folks

    Mu_age_median[i,j] = median(current_sub$EV) / total_folks
    Mu_F_age_median[i,j] = median(current_sub$EV_female) / total_folks
    Mu_M_age_median[i,j] = median(current_sub$EV_male) / total_folks

    # ave
    Mu_age_log_ave[i,j] = mean(current_sub$EV_log_expected) / total_folks
    Mu_F_age_log_ave[i,j] = mean(current_sub$EV_female_log_expected) / total_folks
    Mu_M_age_log_ave[i,j] = mean(current_sub$EV_male_log_expected) / total_folks

    Mu_age_ave[i,j] = mean(current_sub$EV) / total_folks
    Mu_F_age_ave[i,j] = mean(current_sub$EV_female) / total_folks
    Mu_M_age_ave[i,j] = mean(current_sub$EV_male) / total_folks

  }
}


x_age=c(1:length(age_grid))


smooth_N_Age <- kernel2dsmooth(N_Age, kernel.type="disk", r=2)

# self
smooth_Mu_age <- kernel2dsmooth(Mu_age, kernel.type="disk", r=2)
smooth_Mu_M_age <- kernel2dsmooth(Mu_M_age, kernel.type="disk", r=2)
```

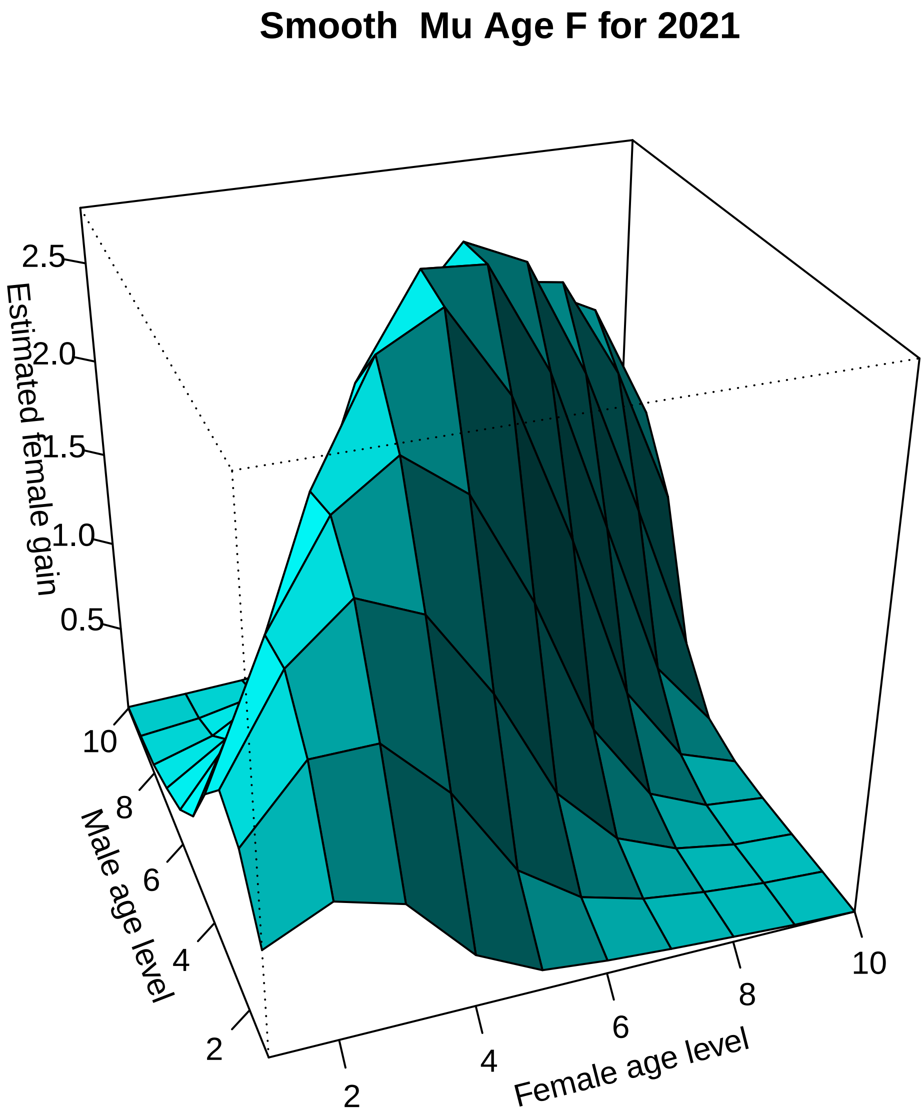
```r
smooth_Mu_F_age <- kernel2dsmooth(Mu_F_age, kernel.type="disk", r=2)

#log
smooth_Mu_age_log <- kernel2dsmooth(Mu_age_log, kernel.type="disk", r=2)
smooth_Mu_M_age_log <- kernel2dsmooth(Mu_M_age_log, kernel.type="disk", r=2)
smooth_Mu_F_age_log <- kernel2dsmooth(Mu_F_age_log, kernel.type="disk", r=2)

# median self
smooth_Mu_age_median <- kernel2dsmooth(Mu_age_median, kernel.type="disk", r=2)
smooth_Mu_M_age_median <- kernel2dsmooth(Mu_M_age_median, kernel.type="disk", r=2)
smooth_Mu_F_age_median <- kernel2dsmooth(Mu_F_age_median, kernel.type="disk", r=2)

# median log
smooth_Mu_age_log_median <- kernel2dsmooth(Mu_age_log_median, kernel.type="disk", r=2)
smooth_Mu_M_age_log_median <- kernel2dsmooth(Mu_M_age_log_median, kernel.type="disk", r=2)
smooth_Mu_F_age_log_median <- kernel2dsmooth(Mu_F_age_log_median, kernel.type="disk", r=2)

#mean self
smooth_Mu_age_ave <- kernel2dsmooth(Mu_age_ave, kernel.type="disk", r=2)
smooth_Mu_M_age_ave <- kernel2dsmooth(Mu_M_age_ave, kernel.type="disk", r=2)
smooth_Mu_F_age_ave <- kernel2dsmooth(Mu_F_age_ave, kernel.type="disk", r=2)

#mean log
smooth_Mu_age_log_ave <- kernel2dsmooth(Mu_age_log_ave, kernel.type="disk", r=2)
smooth_Mu_M_age_log_ave <- kernel2dsmooth(Mu_M_age_log_ave, kernel.type="disk", r=2)
smooth_Mu_F_age_log_ave <- kernel2dsmooth(Mu_F_age_log_ave, kernel.type="disk", r=2)


persp(x_age,x_age,smooth_Mu_age, theta=-20, phi=30, r=5,
      shade=0.4, axes=TRUE,scale=TRUE, box=TRUE,
      nticks=5, ticktype="detailed",
      col="cyan", xlab="Female age level",
      ylab="Male age level", zlab="Estimated mutual gain",
      main=paste("Smooth Mu Age Both for", n_year))
```



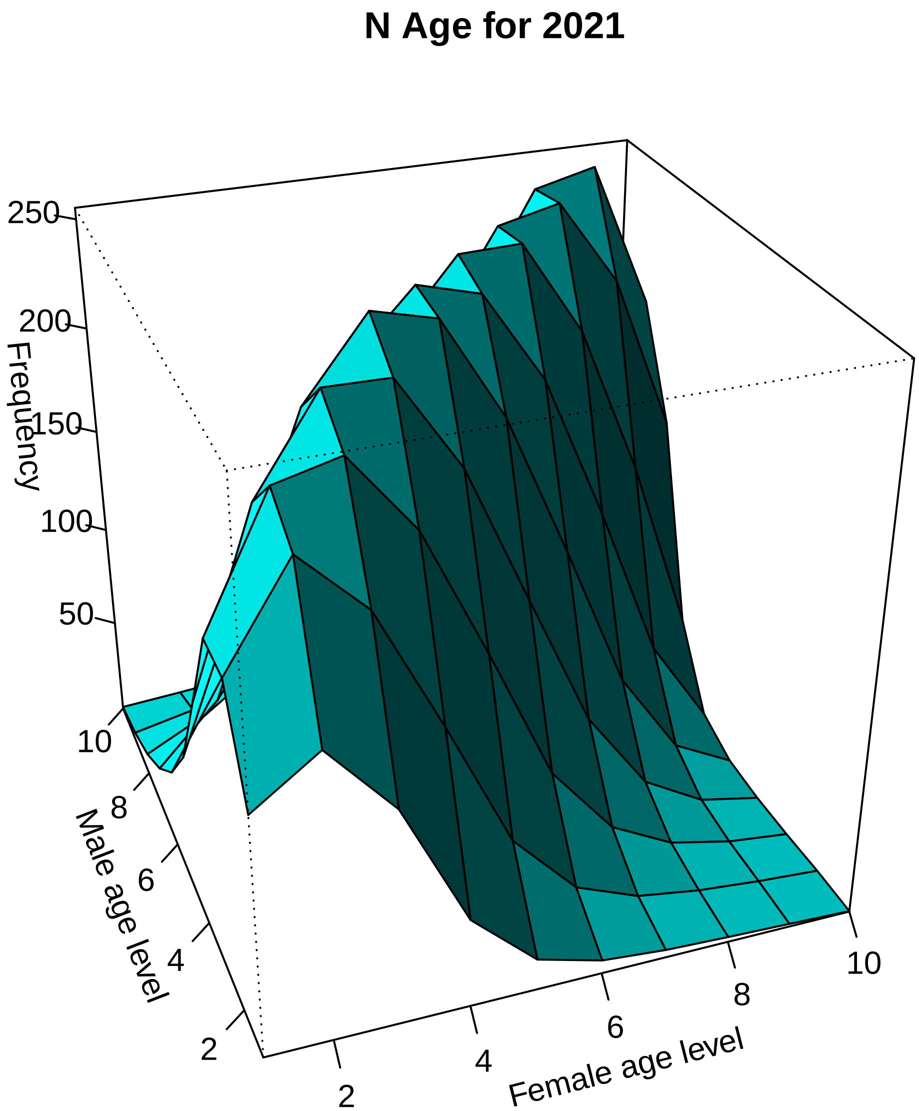

```
persp(x_age,x_age,smooth_Mu_M_age, theta=-20, phi=30, r=5,
      shade=0.4, axes=TRUE,scale=TRUE, box=TRUE,
      nticks=5, ticktype="detailed",
      col="cyan", xlab="Female age level",
      ylab="Male age level", zlab="Estimated male gain",
      main=paste("Smooth Mu Age M for", n_year))
```




```r
persp(x_age,x_age,smooth_Mu_F_age, theta=-20, phi=30, r=5,
      shade=0.4, axes=TRUE,scale=TRUE, box=TRUE,
      nticks=5, ticktype="detailed",
      col="cyan", xlab="Female age level",
      ylab="Male age level", zlab="Estimated female gain",
      main=paste("Smooth  Mu Age F for", n_year))
```

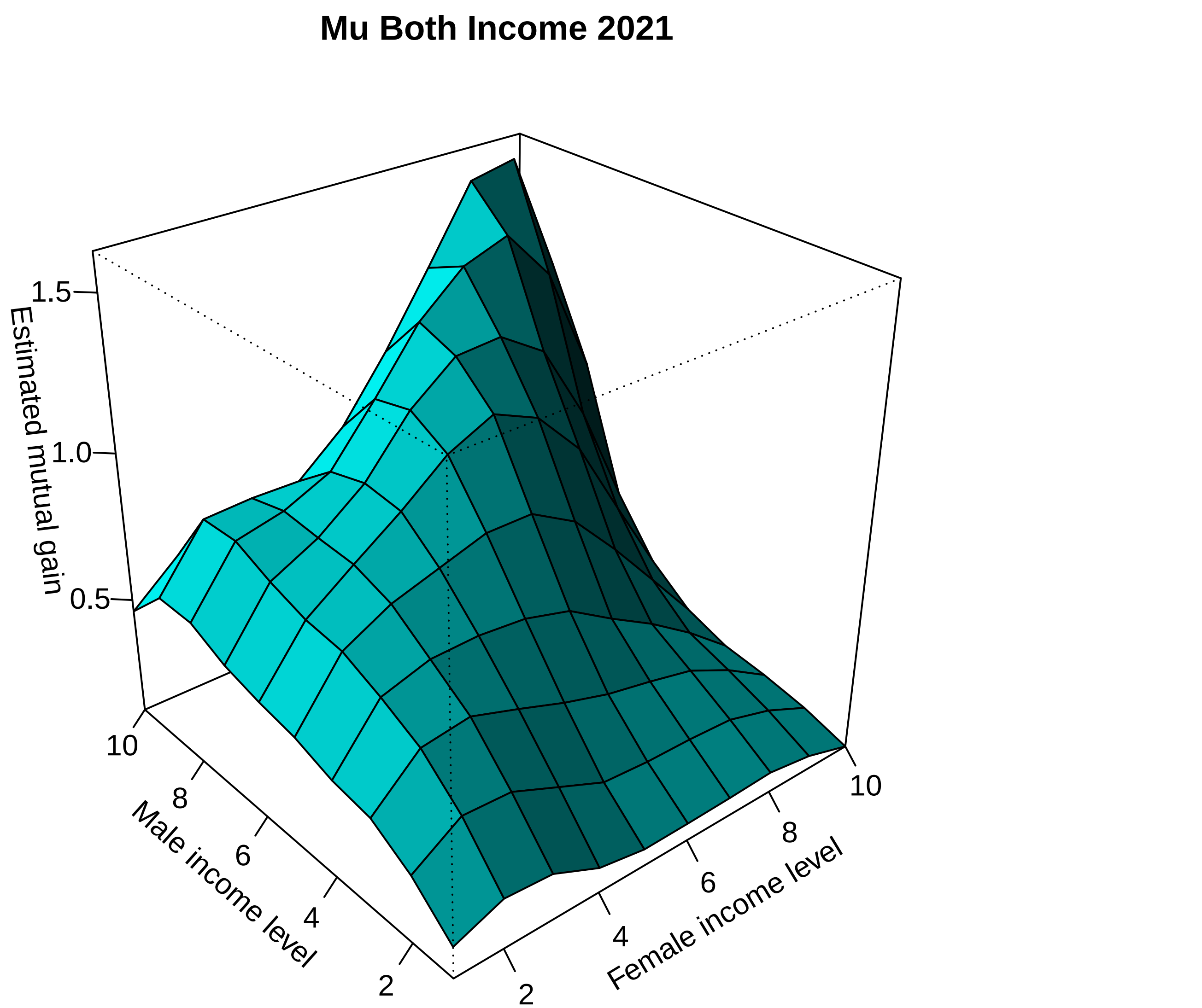



```r
persp(x_age,x_age,smooth_N_Age, theta=-20, phi=30, r=5,
      shade=0.4, axes=TRUE, scale=TRUE, box=TRUE,
      nticks=5, ticktype="detailed",
      col="cyan", xlab="Female age level",
      ylab="Male age level", zlab="Frequency",
      main=paste("N Age for", n_year))
```

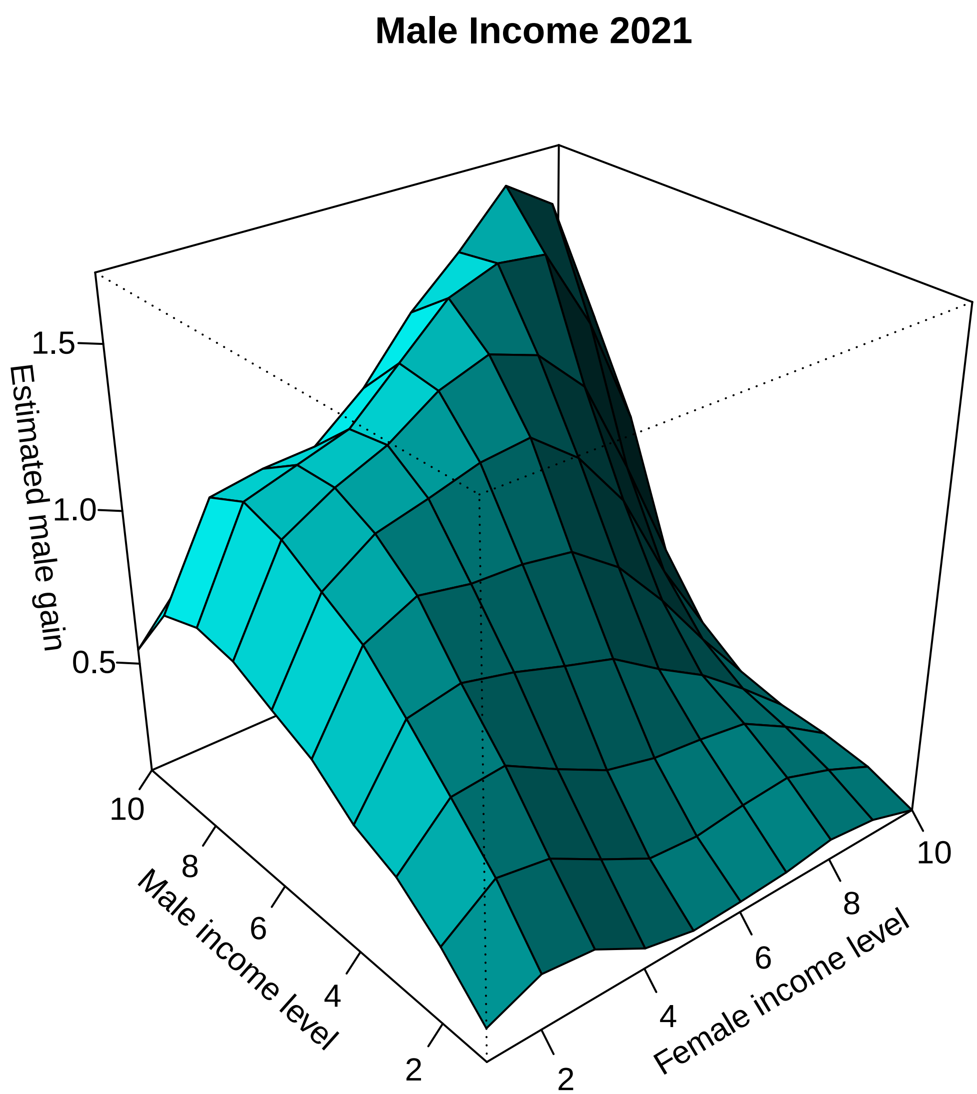



```
####            INCOME

N_income <- matrix(0, n_income, n_income)
Mu_income <- matrix(0, n_income, n_income)
Mu_M_income <- matrix(0, n_income, n_income)
Mu_F_income <- matrix(0, n_income, n_income)
#log version
Mu_income_log<- matrix(0, n_income, n_income)
Mu_M_income_log <- matrix(0, n_income, n_income)
Mu_F_income_log <- matrix(0, n_income, n_income)

for (i in 1:n_income){
  for (j in 1:n_income){
    #use pi
    current_sub <- subset(Mu_all, Mu_all$NEW_INCTOT.x == i & Mu_all$NEW_INCTOT.y == j)

    total_folks = 1 # sum(current_sub$n)

    Mu_income_log[i,j] = sum(current_sub$EV_log) / total_folks
```

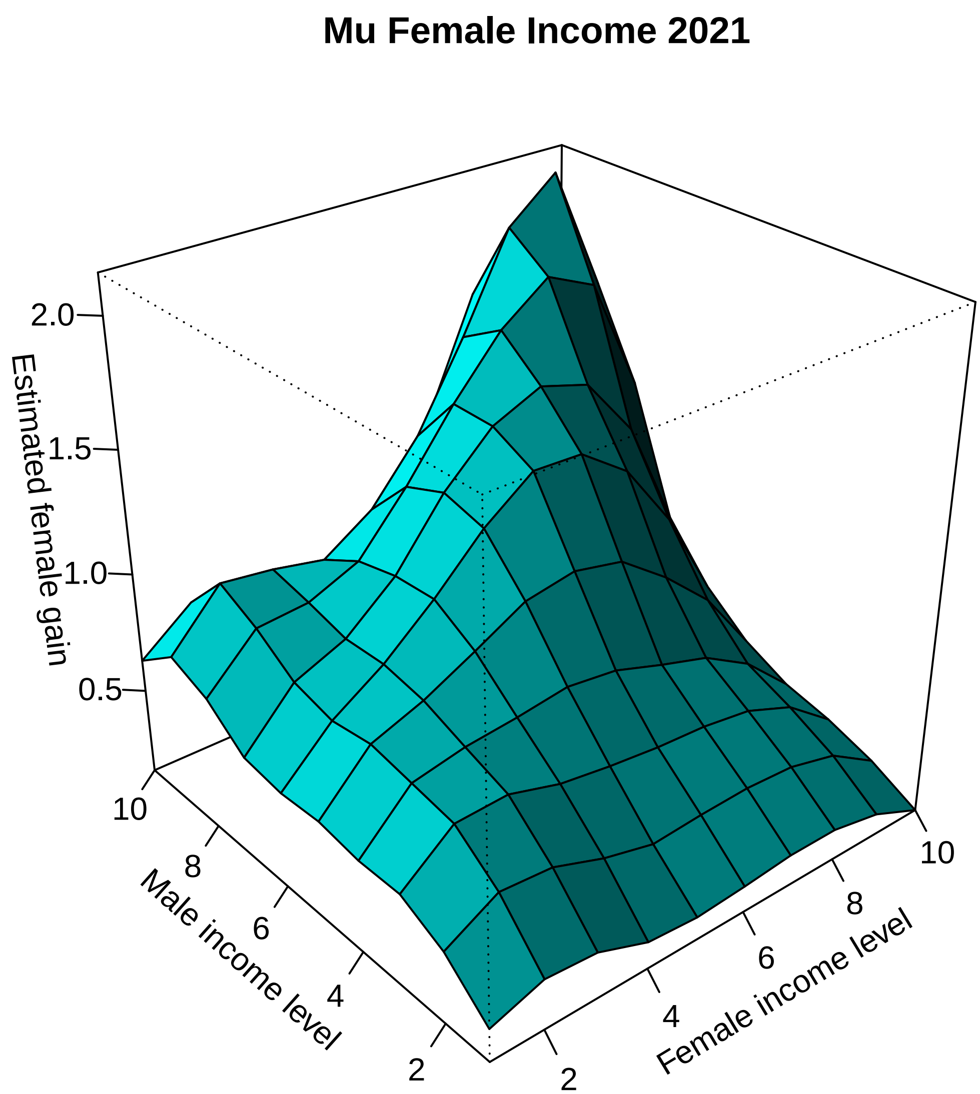
```r
    Mu_F_income_log[i,j] = sum(current_sub$EV_female_log) / total_folks
    Mu_M_income_log[i,j] = sum(current_sub$EV_male_log) / total_folks


    Mu_income[i,j] = sum(current_sub$EV) / total_folks
    Mu_F_income[i,j] = sum(current_sub$EV_female) / total_folks
    Mu_M_income[i,j] = sum(current_sub$EV_male) / total_folks
    N_income[i,j] = dim(current_sub)[1]

  }
}

x_income=c(1:length(income_grid))


smooth_Mu_income <- kernel2dsmooth(Mu_income, kernel.type="disk", r=2)
smooth_Mu_M_income <- kernel2dsmooth(Mu_M_income, kernel.type="disk", r=2)
smooth_Mu_F_income <- kernel2dsmooth(Mu_F_income, kernel.type="disk", r=2)

smooth_N_income <- kernel2dsmooth(N_income, kernel.type="disk", r=2)

#theta=-20, phi=30, r=35,
persp(x_income,x_income,smooth_Mu_income, theta=-40, phi=30, r=5,#theta=-20, phi=30, r=35, #theta=120, 
      shade=0.6, axes=TRUE,scale=TRUE, box=TRUE,
      nticks=5, ticktype="detailed",
      col="cyan", xlab="Female income level",
      ylab="Male income level", zlab="Estimated mutual gain",
      main=paste("Mu Both Income", n_year))
```



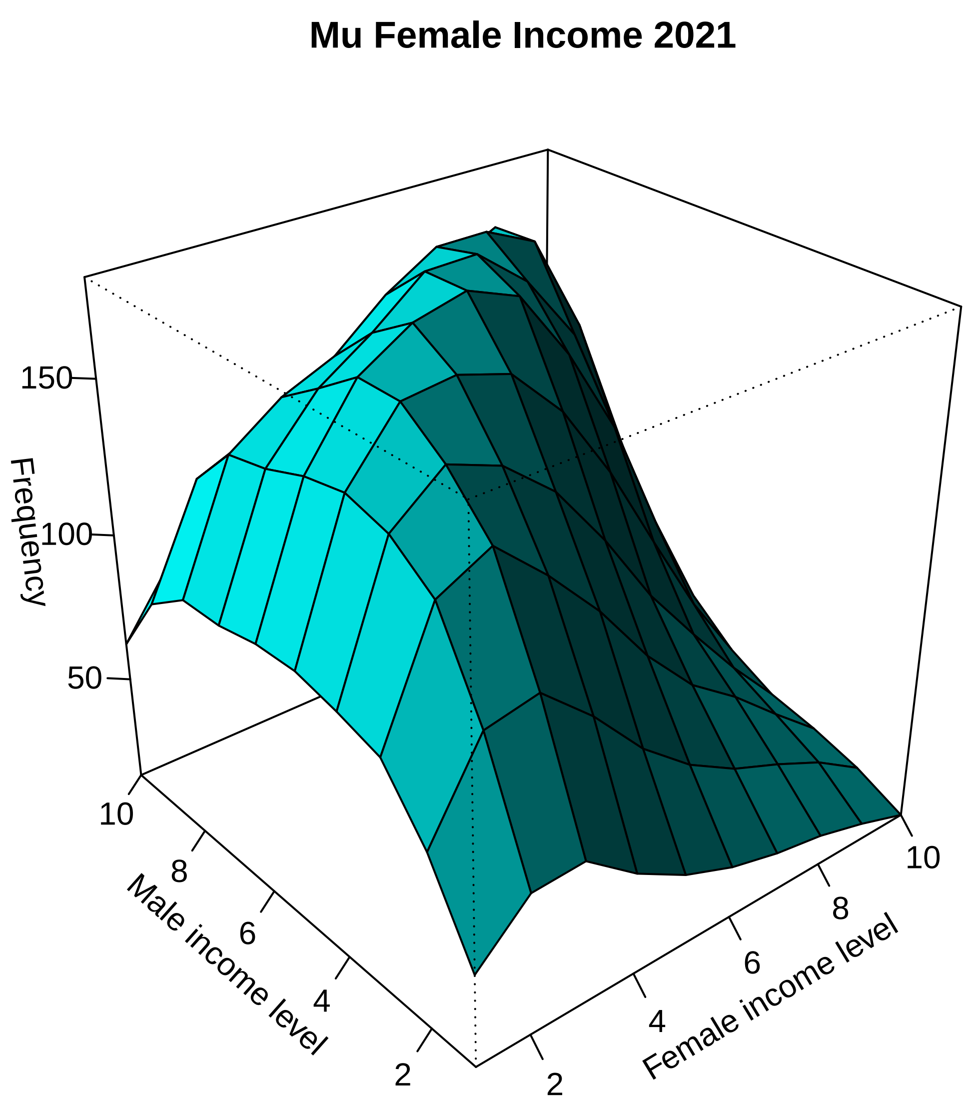

```
persp(x_income,x_income,smooth_Mu_M_income, theta=-40, phi=30, r=5, #theta=120, phi=30, r=35,
      shade=0.6, axes=TRUE,scale=TRUE, box=TRUE,
      nticks=5, ticktype="detailed",
      col="cyan", xlab="Female income level",
      ylab="Male income level", zlab="Estimated male gain",
      main=paste("Male Income", n_year))
```




```
persp(x_income,x_income,smooth_Mu_F_income, theta=-40, phi=30, r=5, #theta=120, phi=30, r=35,
      shade=0.6, axes=TRUE,scale=TRUE, box=TRUE,
      nticks=5, ticktype="detailed",
      col="cyan", xlab="Female income level",
      ylab="Male income level", zlab="Estimated female gain",
      main=paste("Mu Female Income", n_year))
```

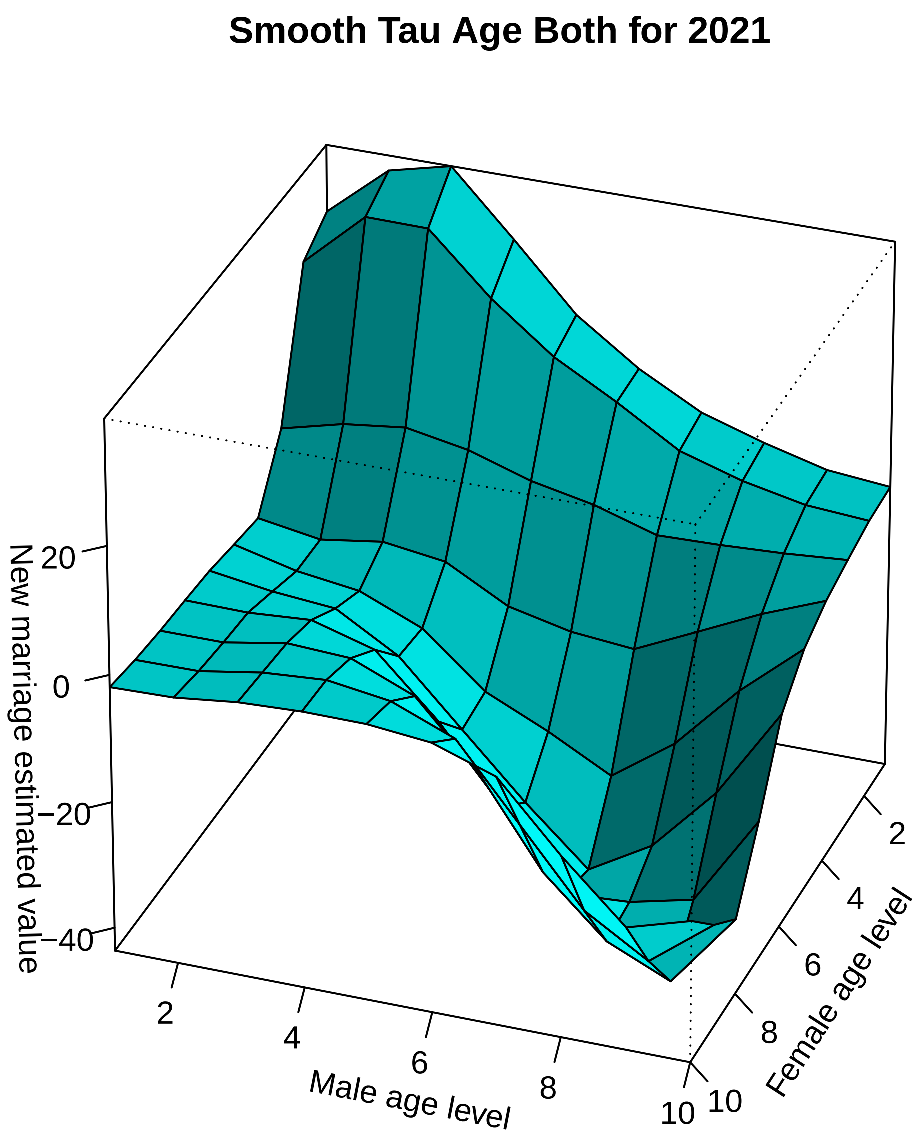



```
persp(x_income,x_income,smooth_N_income, theta=-40, phi=30, r=5, #theta=120, phi=30, r=35,
      shade=0.6, axes=TRUE,scale=TRUE, box=TRUE,
      nticks=5, ticktype="detailed",
      col="cyan", xlab="Female income level",
      ylab="Male income level", zlab="Frequency",
      main=paste("Mu Female Income", n_year))
```



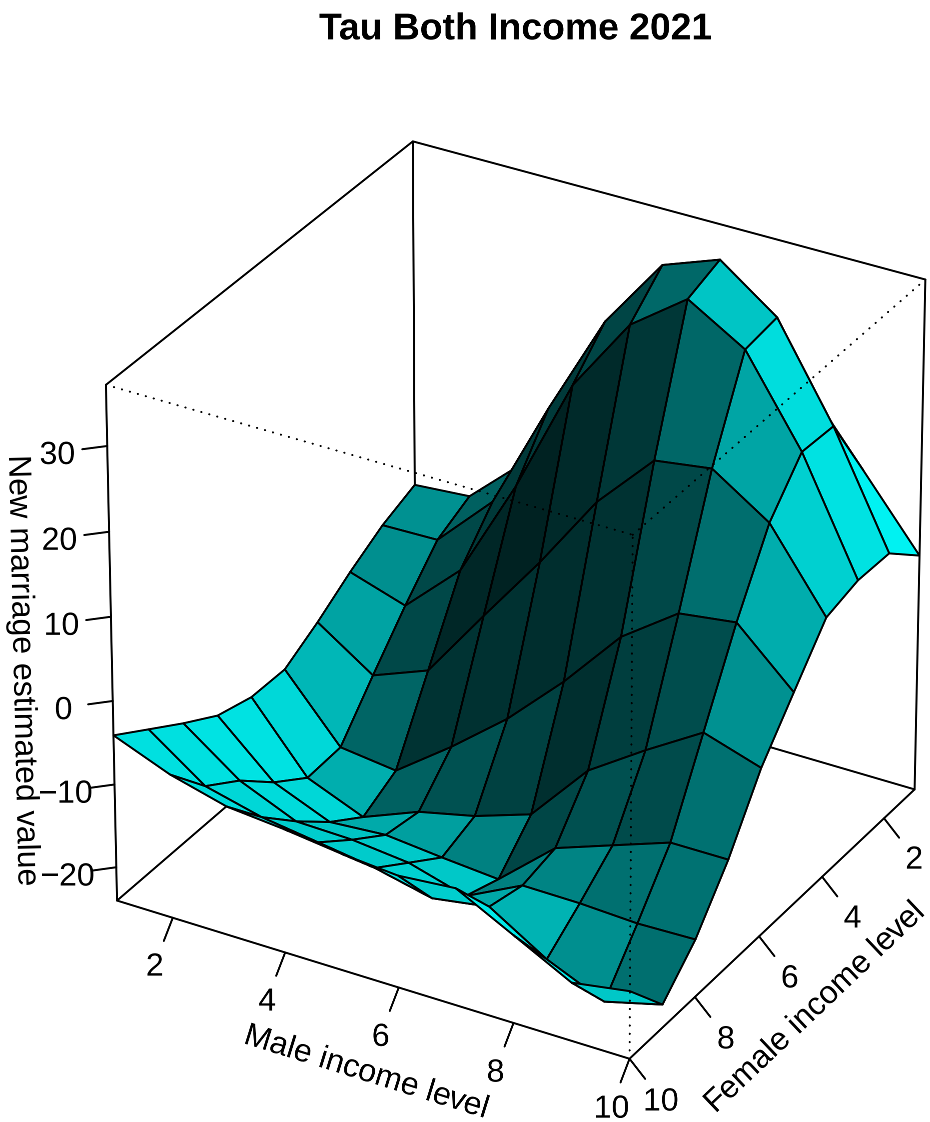

```
####            EDUCATION

N_edu <- matrix(0, n_edu, n_edu)
Mu_edu <- matrix(0, n_edu, n_edu)
Mu_M_edu <- matrix(0, n_edu, n_edu)
Mu_F_edu <- matrix(0, n_edu, n_edu)
#log version
Mu_edu_log<- matrix(0, n_edu, n_edu)
Mu_M_edu_log <- matrix(0, n_edu, n_edu)
Mu_F_edu_log <- matrix(0, n_edu, n_edu)

for (i in 1:n_edu){
  for (j in 1:n_edu){
    #use pi
    current_sub <- subset(Mu_all, Mu_all$NEW_EDUCD.x == i & Mu_all$NEW_EDUCD.y == j)

    total_folks = 1 # sum(current_sub$n)

    Mu_edu_log[i,j] = sum(current_sub$EV_log) / total_folks
```

```r
    Mu_F_edu_log[i,j] = sum(current_sub$EV_female_log) / total_folks
    Mu_M_edu_log[i,j] = sum(current_sub$EV_male_log) / total_folks


    Mu_edu[i,j] = sum(current_sub$EV) / total_folks
    Mu_F_edu[i,j] = sum(current_sub$EV_female) / total_folks
    Mu_M_edu[i,j] = sum(current_sub$EV_male) / total_folks
    N_edu[i,j] = dim(current_sub)[1]

  }
}

x_edu=c(1:length(edu_grid))


smooth_Mu_edu <- kernel2dsmooth(Mu_edu, kernel.type="disk", r=2)
smooth_Mu_M_edu <- kernel2dsmooth(Mu_M_edu, kernel.type="disk", r=2)
smooth_Mu_F_edu <- kernel2dsmooth(Mu_F_edu, kernel.type="disk", r=2)


#tau
Tau_age <- matrix(0, n_age, n_age)

for (i in 1:n_age){
  for (j in 1:n_age){
    current_sub <- subset(Mu_all, Mu_all$NEW_AGE.x == i & Mu_all$NEW_AGE.y == j)
    total_folks =  1 # sum(current_sub$n)
    Tau_age[i,j] = sum(current_sub$tau) / total_folks
  }
}

x_age=c(1:length(age_grid))
smooth_Tau_age <- kernel2dsmooth(Tau_age, kernel.type="disk", r=2)

persp(x_age,x_age,smooth_Tau_age, theta=110, phi=30, r=35,
      shade=0.4, axes=TRUE,scale=TRUE, box=TRUE,
      nticks=5, ticktype="detailed",
      col="cyan", xlab="Female age level",
      #zlim = c(0,2*max(smooth_Mu_age[!is.na(smooth_Mu_age)])),
      ylab="Male age level", zlab="New marriage estimated value",
      main=paste("Smooth Tau Age Both for", n_year))
```




```
Tau_income <- matrix(0, n_income, n_income)

for (i in 1:n_income){
  for (j in 1:n_income){
    #use pi
    current_sub <- subset(Mu_all, Mu_all$NEW_INCTOT.x == i & Mu_all$NEW_INCTOT.y == j)

    total_folks = 1 # sum(current_sub$n)

    Tau_income[i,j] = sum(current_sub$tau) / total_folks
  }
}

x_income=c(1:length(income_grid))
smooth_Tau_income <- kernel2dsmooth(Tau_income, kernel.type="disk", r=2)
persp(x_income,x_income,smooth_Tau_income, theta=120, phi=30, r=35,
      shade=0.6, axes=TRUE,scale=TRUE, box=TRUE,
      nticks=5, ticktype="detailed",
```

```
      col="cyan", xlab="Female income level",
      ylab="Male income level", zlab="New marriage estimated value",
      main=paste("Tau Both Income", n_year))
```



Explanation of R Code for Data Retrieval

The following code snippet retrieves data from a function named `funk`, which is assumed to estimate marriage demand values for the year 2020. The function returns a list containing various elements, out of which `Mu_all` and `pair_data` are extracted.

```
Mu_all <- funk(2020)$Mu_all
pair_data <- funk(2020)$pair_data
```

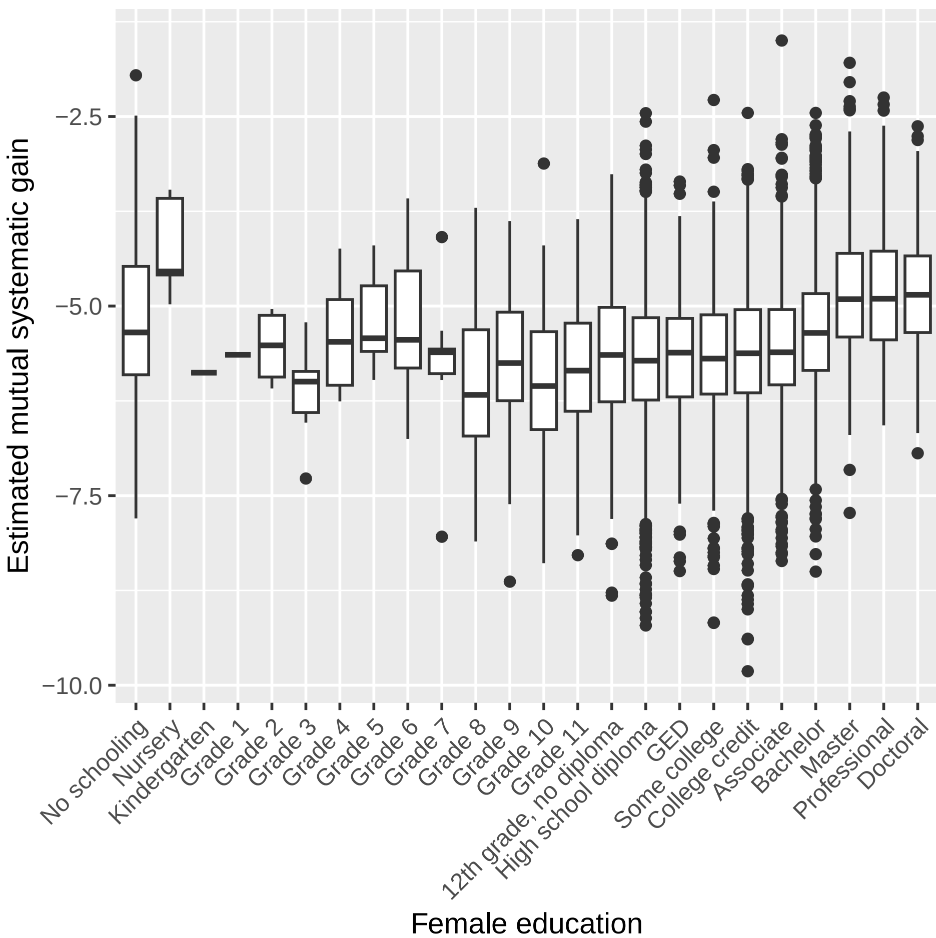
R Code Explanation: Data Visualization using ggplot2

Plotting Education-based Mutual Systematic Gain with Boxplots

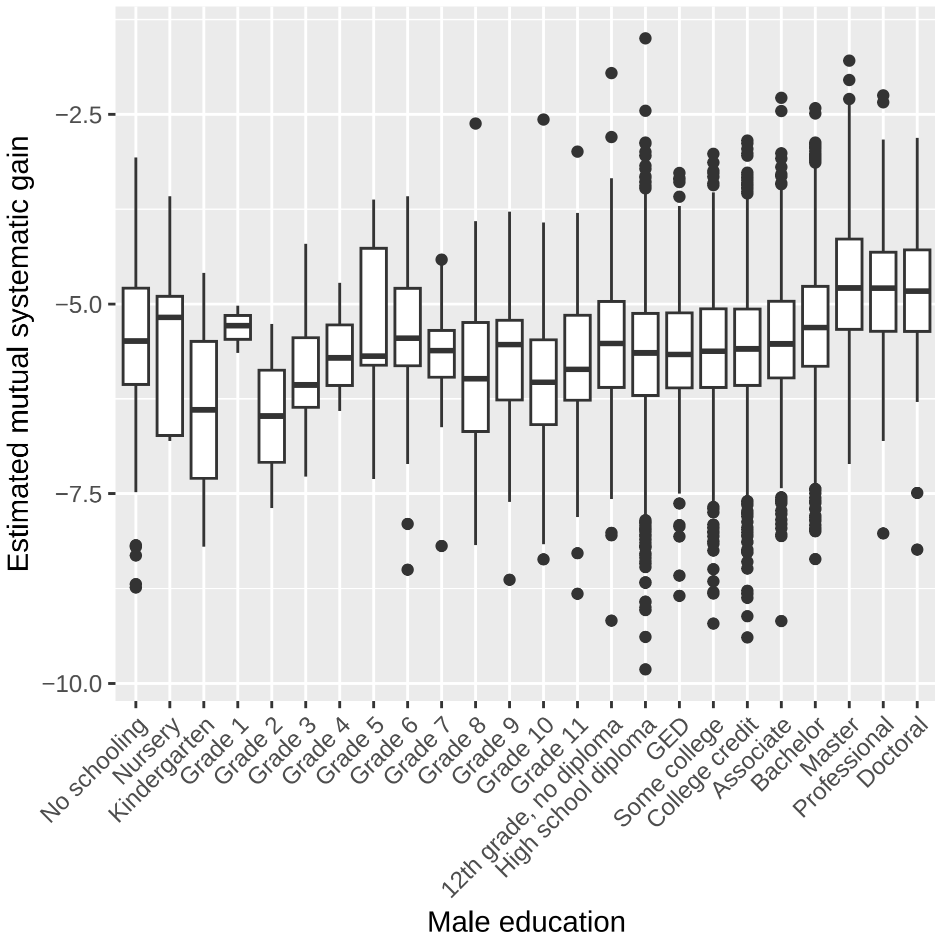
The following code block creates boxplots to visualize the distribution of estimated mutual systematic gain across different education levels for both males and females.

Code for creating boxplots based on education

```
# EDUCD with boxplot

ggplot(pair_data, aes(x = factor(EDUCD.x), y = MV_log)) +
  geom_boxplot() + scale_x_discrete(labels=edu_labels) +
  xlab("Female education") + ylab("Estimated mutual systematic gain") +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust = 1))
```

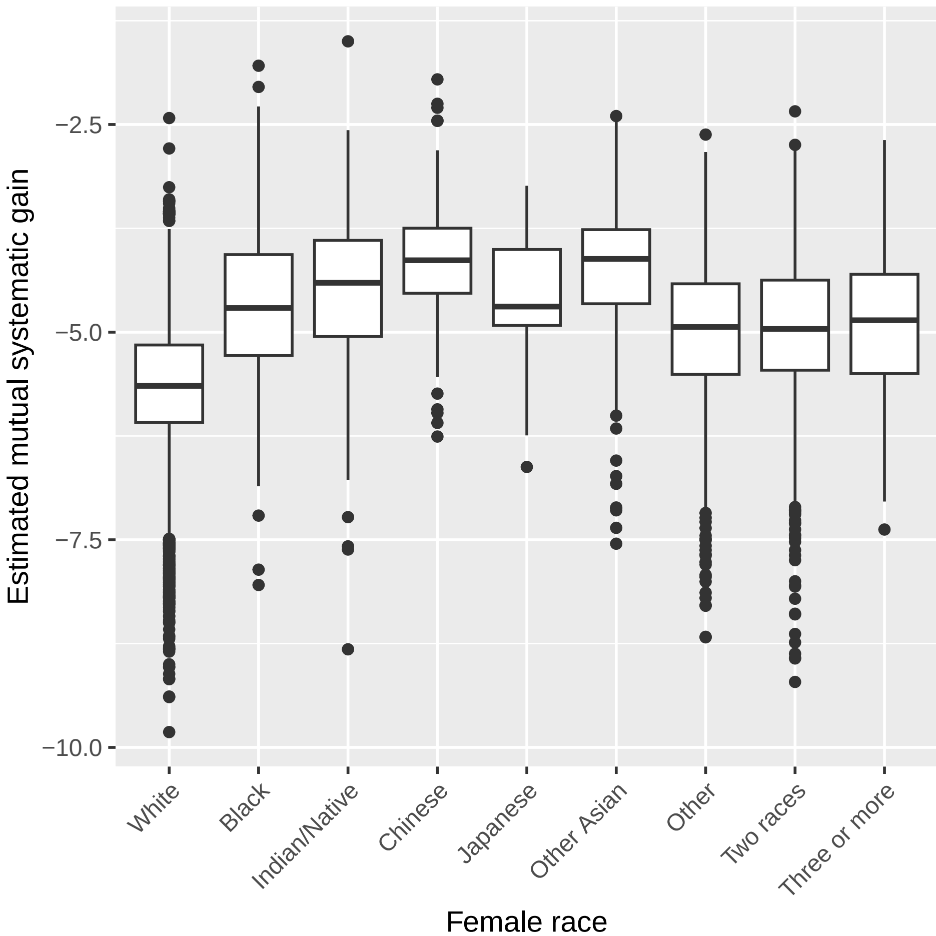



```
ggplot(pair_data, aes(x = factor(EDUCD.y), y = MV_log)) +
  geom_boxplot() + scale_x_discrete(labels=edu_labels) +
  xlab("Male education") + ylab("Estimated mutual systematic gain") +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust = 1))
```

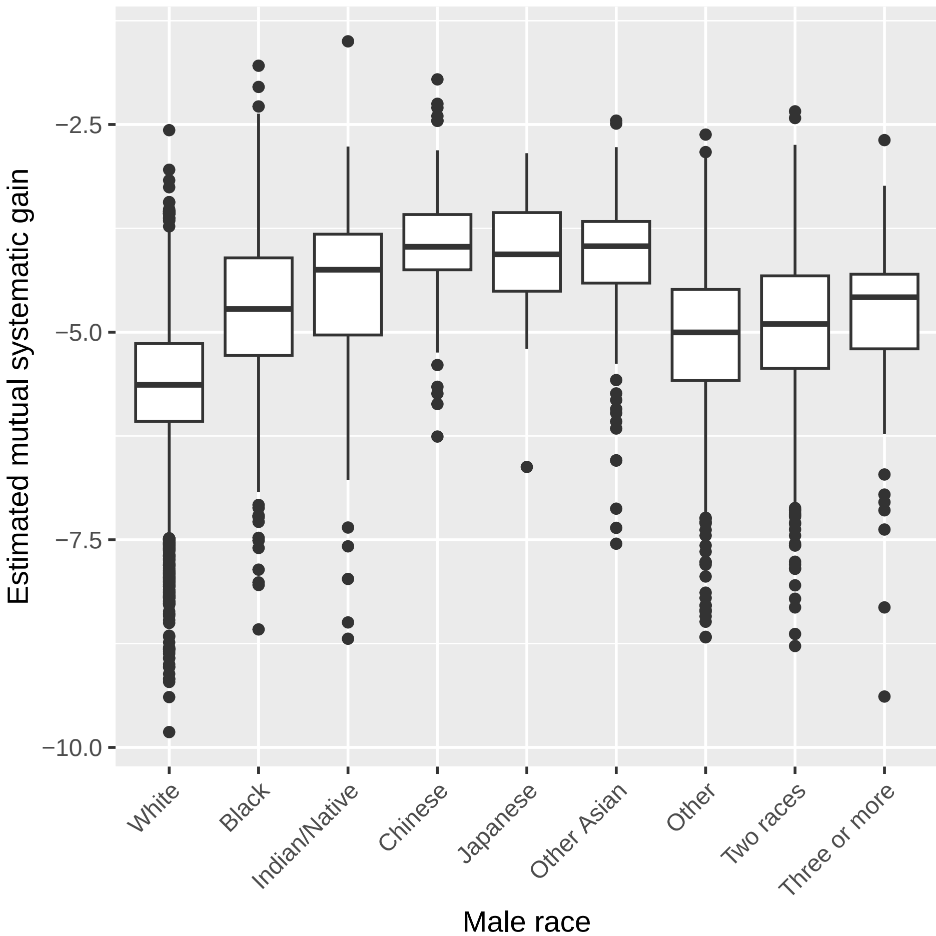


Code for creating boxplots based on race

```
# RACE with boxplot

ggplot(pair_data, aes(x = factor(RACE.x), y = MV_log)) +
  geom_boxplot() + scale_x_discrete(labels=race_labels) +
  xlab("Female race") + ylab("Estimated mutual systematic gain") +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust = 1))
```



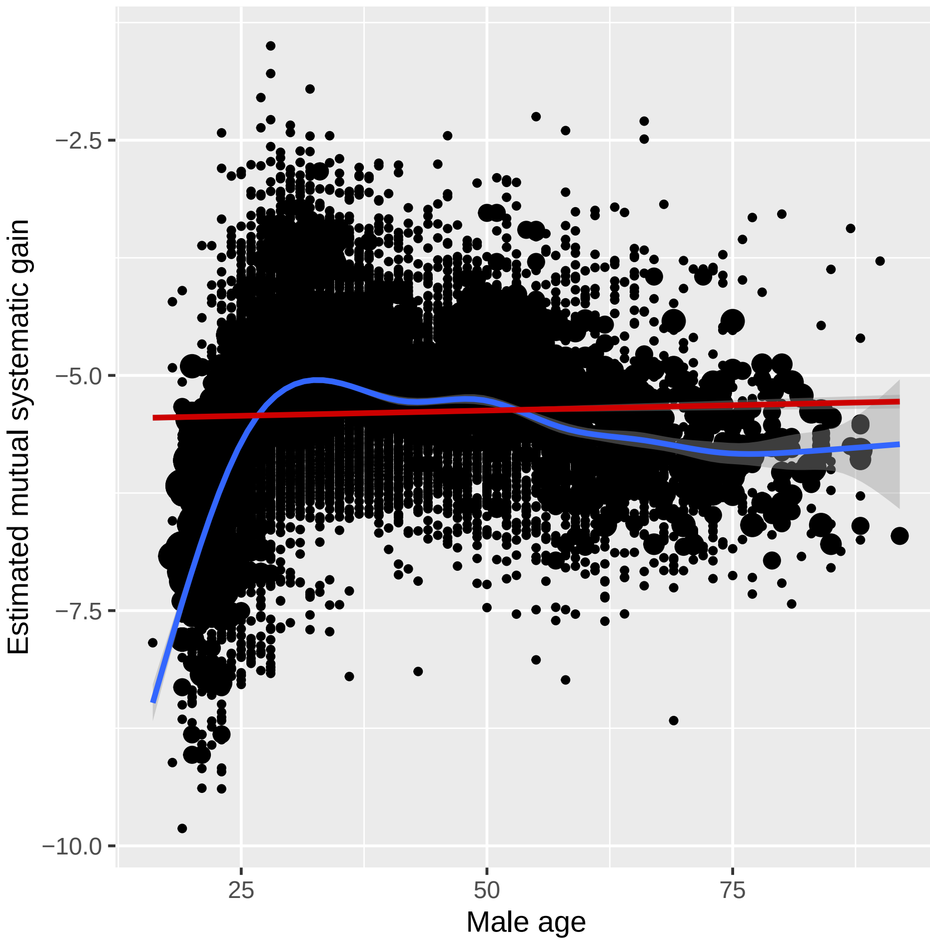

```
ggplot(pair_data, aes(x = factor(RACE.y), y = MV_log)) +
  geom_boxplot() + scale_x_discrete(labels=race_labels) +
  xlab("Male race") + ylab("Estimated mutual systematic gain") +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust = 1))
```



Code for creating scatter and smooth plots based on age

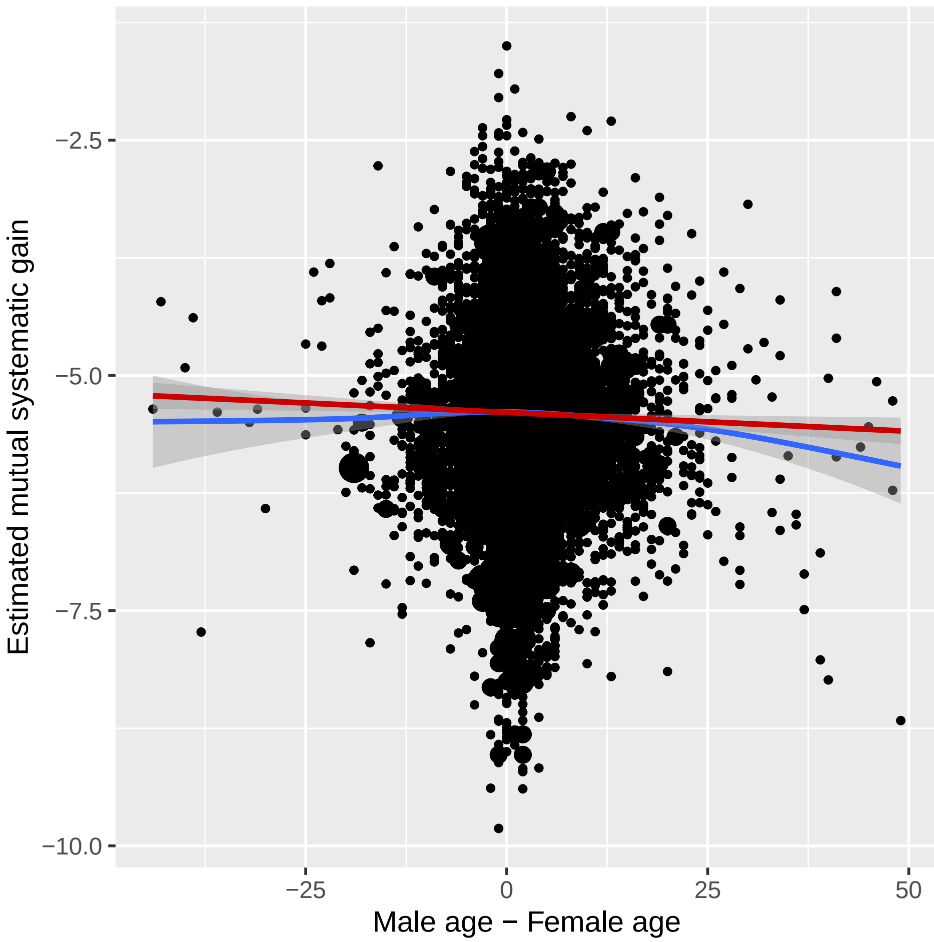
```
ggplot(data= pair_data,aes(x=AGE.y,y=MV_log)) +
  geom_point(aes(size=n)) +
  geom_smooth(weight="HHWT.x",show.legend = FALSE) +
  geom_smooth(method = "lm", weight="HHWT.x", color="Red3", show.legend = FALSE) +
  theme(legend.position="none") +
  xlab("Male age") +
  ylab("Estimated mutual systematic gain")
```

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using formula = 'y ~ x'
```




```r
ggplot(data= pair_data,aes(x=AGE.y-AGE.x,y=MV_log)) +
  geom_point(aes(size=n)) +
  geom_smooth(weight="HHWT.x",show.legend = FALSE) +
  geom_smooth(method = "lm", weight="HHWT.x", color="Red3", show.legend = FALSE) +
  theme(legend.position="none") +
  xlab("Male age - Female age") +
  ylab("Estimated mutual systematic gain")
```

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using formula = 'y ~ x'
```



Analysis of Influences on Mutual Gain in Marriage

d In this section, we investigate how various factors like education, income, and age of both partners influence their mutual gain in marriage. For this purpose, we run a linear model using the `lm()` function.

```
store_jm <- funk(2021)
pair_data_jm <- store_jm$pair_data


# Running linear model to examine influence of various factors on mutual gain in marriage


fit_jm_int <- lm(1000*MV_log ~
               as.integer(NEW_AGE.x) + as.integer(NEW_AGE.y)
             + as.integer(NEW_INCTOT.x) + as.integer(NEW_INCTOT.y)
             + as.integer(NEW_EDUCD.x) + as.integer(NEW_EDUCD.y)
             + factor(NEW_RACE.x) + factor(NEW_RACE.y)
             , data = pair_data_jm
             , weights = pair_data_jm$HHW)

summary(fit_jm_int)
```

```
##
## Call:
```

```
## lm(formula = 1000 * MV_log ~ as.integer(NEW_AGE.x) + as.integer(NEW_AGE.y) +
##     as.integer(NEW_INCTOT.x) + as.integer(NEW_INCTOT.y) + as.integer(NEW_EDUCD.x) +
##     as.integer(NEW_EDUCD.y) + factor(NEW_RACE.x) + factor(NEW_RACE.y),
##     data = pair_data_jm, weights = pair_data_jm$HHW)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3206.1  -354.6     9.8   394.3  2585.7
##
## Coefficients:
##                           Estimate Std. Error  t value Pr(>|t|)
## (Intercept)              -7295.787     20.285 -359.669  < 2e-16 ***
## as.integer(NEW_AGE.x)       24.041      3.625    6.632 3.44e-11 ***
## as.integer(NEW_AGE.y)        2.950      3.704    0.796    0.426
## as.integer(NEW_INCTOT.x)    53.479      2.148   24.893  < 2e-16 ***
## as.integer(NEW_INCTOT.y)    77.895      2.160   36.062  < 2e-16 ***
## as.integer(NEW_EDUCD.x)    111.719      6.387   17.492  < 2e-16 ***
## as.integer(NEW_EDUCD.y)    104.802      6.182   16.952  < 2e-16 ***
## factor(NEW_RACE.x)2        387.472     33.506   11.564  < 2e-16 ***
## factor(NEW_RACE.x)3        675.133     25.157   26.837  < 2e-16 ***
## factor(NEW_RACE.x)4        349.639     16.467   21.233  < 2e-16 ***
## factor(NEW_RACE.y)2        713.124     31.254   22.817  < 2e-16 ***
## factor(NEW_RACE.y)3        821.163     27.820   29.517  < 2e-16 ***
## factor(NEW_RACE.y)4        439.857     16.826   26.141  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 618.6 on 14724 degrees of freedom
## Multiple R-squared:  0.485,  Adjusted R-squared:  0.4845
## F-statistic:  1155 on 12 and 14724 DF,  p-value: < 2.2e-16
```