R Notebook

This is an R Markdown (http://rmarkdown.rstudio.com) Notebook. When you execute code within the notebook, the results appear beneath the code.

```
#initial stuff
rm(list = ls())
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.1.1     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
#see all the digits (hopefully)
options(digits = 22)
memory.limit(20000)
```

```
## [1] 20000
```

```
setwd("E:/Duke/Semester 2/Applied Micro/Assignments/A1")
getwd()
```

```
## [1] "E:/Duke/Semester 2/Applied Micro/Assignments/A1"
```

```
#loading in data
# find all files and then import them separately
dataFiles = list.files("E:/Duke/Semester 2/Applied Micro/Assignments/A1",pattern="*.csv") #all d
ata should be .csv files so this should get all the names (including .csv extension)
# for (i in 1:(length(dataFiles)/2)) assign(dataFiles[i], read.csv(dataFiles[i],colClasses = c
("idmen" = "character"),header = TRUE))
# for (i in 1+(length(dataFiles)/2):length(dataFiles)) assign(dataFiles[i], read.csv(dataFiles
[i],colClasses = c("idind" ="character","idmen" = "character"),header = TRUE))

for (i in 1:length(dataFiles)) assign(dataFiles[i], read.csv(dataFiles[i],colClasses = c("idind"
="character","idmen" = "character"),header = TRUE))
```

```
## Warning in read.table(file = file, header = header, sep = sep, quote = quote, :
## not all columns named in 'colClasses' exist

## Warning in read.table(file = file, header = header, sep = sep, quote = quote, :
## not all columns named in 'colClasses' exist

## Warning in read.table(file = file, header = header, sep = sep, quote = quote, :
## not all columns named in 'colClasses' exist

## Warning in read.table(file = file, header = header, sep = sep, quote = quote, :
## not all columns named in 'colClasses' exist

## Warning in read.table(file = file, header = header, sep = sep, quote = quote, :
## not all columns named in 'colClasses' exist

## Warning in read.table(file = file, header = header, sep = sep, quote = quote, :
## not all columns named in 'colClasses' exist

## Warning in read.table(file = file, header = header, sep = sep, quote = quote, :
## not all columns named in 'colClasses' exist

## Warning in read.table(file = file, header = header, sep = sep, quote = quote, :
## not all columns named in 'colClasses' exist

## Warning in read.table(file = file, header = header, sep = sep, quote = quote, :
## not all columns named in 'colClasses' exist

## Warning in read.table(file = file, header = header, sep = sep, quote = quote, :
## not all columns named in 'colClasses' exist

## Warning in read.table(file = file, header = header, sep = sep, quote = quote, :
## not all columns named in 'colClasses' exist

## Warning in read.table(file = file, header = header, sep = sep, quote = quote, :
## not all columns named in 'colClasses' exist

## Warning in read.table(file = file, header = header, sep = sep, quote = quote, :
## not all columns named in 'colClasses' exist

## Warning in read.table(file = file, header = header, sep = sep, quote = quote, :
## not all columns named in 'colClasses' exist

## Warning in read.table(file = file, header = header, sep = sep, quote = quote, :
## not all columns named in 'colClasses' exist

## Warning in read.table(file = file, header = header, sep = sep, quote = quote, :
## not all columns named in 'colClasses' exist
```

```
#got warnings about colClasses sometimes not existing - believe that is because idind is not in
 hhDatas

#read all IDs as characters as there were issues with repetition due to R not fully reading valu
es (some truncation issue)

#Get warnings because the hhData doesn't have idind and I just changed all the idmen and idind t
o characters for the issue's discussed in slack
```

Exercise 1

```
#a. Number of Households in 2007
nrow(dathh2007.csv)
```

```
## [1] 10498
```

```
numHH07 <- dathh2007.csv%>%group_by(idmen)%>%summarize(count= n())
sum(numHH07$count)
```

```
## [1] 10498
```

```
#b. Number of households with marital status - Couple with Kids - in 2005
# used table function and just checked when the mstatus was True in 2005
table(dathh2005.csv$mstatus == "Couple, with Kids")
```

```
##
## FALSE  TRUE
##  6379  3374
```

```
#c. Number of individuals surveyed in 2008
#
numIND08 <- datind2008.csv%>%group_by(idind)%>%summarize(count= n())
sum(numIND08$count)
```

```
## [1] 25510
```

```
#each row should be unique person so these two, if data is good, should be the same
nrow(datind2008.csv)
```

```
## [1] 25510
```

```
#d. Number of individuals between 25 and 35 in '16
# made a condition that only selected ages between 25 and 35 and chose that subset of the corres
ponding variable in the right year.
condition <-  datind2016.csv$age >= 25 & datind2016.csv$age <= 35
#length gives us the number of people between those ages
length(datind2016.csv$age[condition])
```

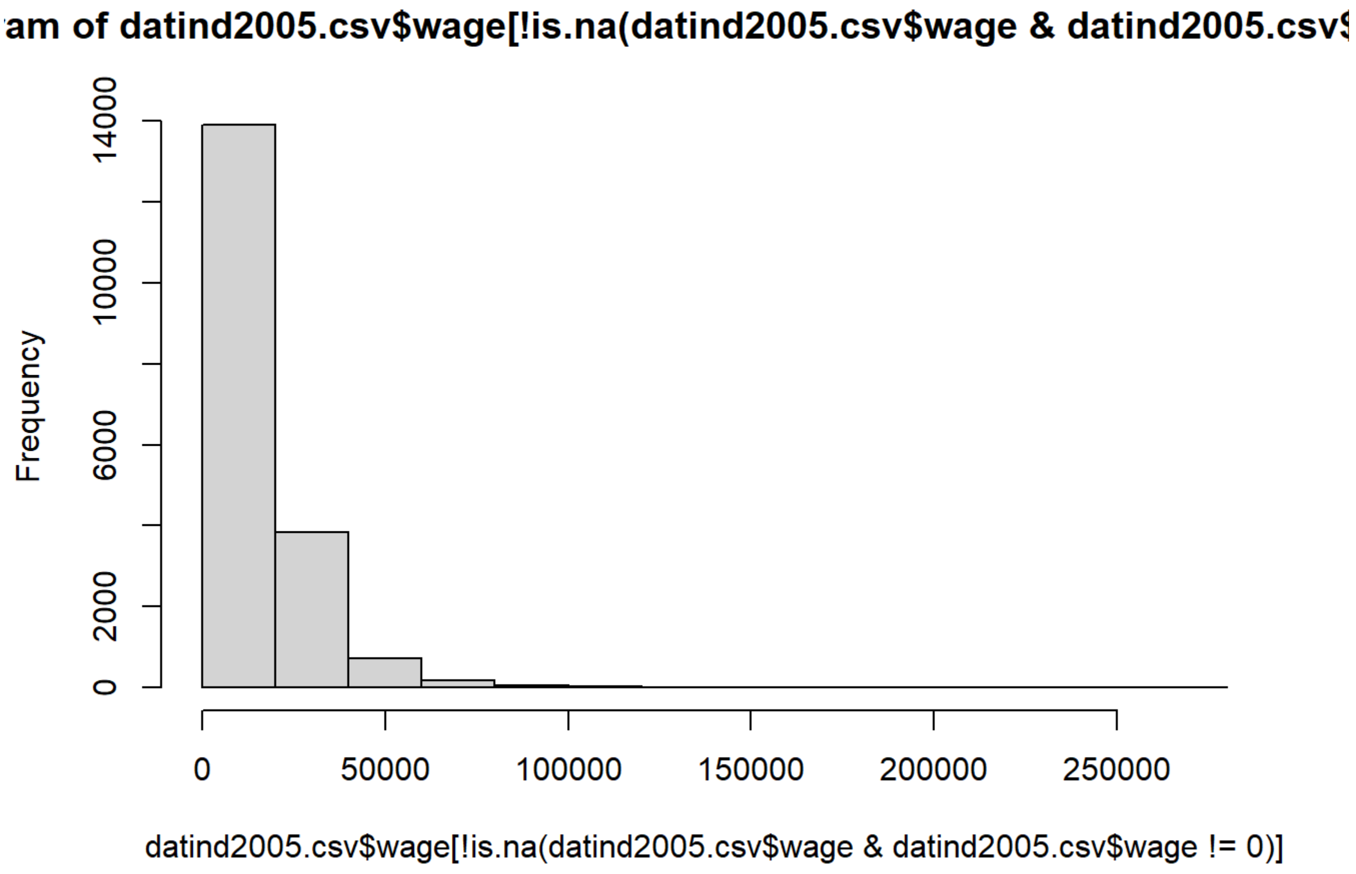
```
## [1] 2765
```

```
#e. Cross-Table Gender/Profession in 2009
table(datind2009.csv$gender,datind2009.csv$profession)
```

```
##         
##            0  11  12  13  21  22  23  31  33  34  35  37  38  42  43  44  45
##   Female  11  30   8  29  63  65   8  68  85 184  50 179  78 258 437   1 153
##   Male    19  57  19  78 213 114  48  98 107 142  59 260 368 110 117   2  95
##         
##           46  47  48  52  53  54  55  56  62  63  64  65  67  68  69
##   Female 410  82  22 782  27 584 353 696  64  35  29  19 147 120  40
##   Male   340 429 215 169 182  98 101  74 443 520 246 159 237 177  82
```

```
#profession is a categorical variable so it will be some numbers
```

```
#f. distribution of wages in 2005 and 2019.

####################
  #calculating Gini
  # for simplicity, I will assume that individuals in each "bucket" earn the same amount
  # ie if, in decile 5, the value is 500 - I assume everybody in the 10% of all make 500.
  # for extra simplicity (scaling), will assume 1 person in each bucket (so in theory, we could
 just scale up all wages but shouldn't change stuff)
  #get deciles (10% - 100%). Also showing Lorenz Curve and Equality Curves

  ####################

  GINI <- function(inputData){
    deciles <- quantile(inputData,seq(.1,1,by=.00001)) # really small increments so we can accur
ately get areas and get corresponding values for deciles


    #extract data into vector
    decile_Vector <- c()
    for (i in 1:length(deciles)){
      decile_Vector <- c(decile_Vector,deciles[[i]])
    }
    #find total income (needed for finding % of total income at an index point)
    totalIncome <- sum(decile_Vector) #finding total

    cumInc <- cumsum(decile_Vector)/totalIncome #find cumulative income at each of the decile le
vels

    #plot for visuals
    plot(cumInc, type = "l",lwd = 3, ylab = "Cum Income Share") #"Lorenz" Curve approximation
    abline(a=0,b=1/length(deciles), lwd = 3, lty = "dotted" ) # full equality line
    # find area between curves ratio

    library(pracma) # needed to find area
    underLorenz <- trapz(1:length(deciles),cumInc) # way to calculate area using trapezoidal app
roximation
    area <- length(deciles)*1*.5
    gini <- (area-underLorenz)/area
    return(gini)

  }

# separately

  #2005
#cannot do certain calculations with missing data so choosing to omit that. Many 0's but keeping
that in as it may be likely that many individuals don't have an income for one reason or another
  hist(datind2005.csv$wage[!is.na(datind2005.csv$wage & datind2005.csv$wage != 0)])
```



```
 mean(datind2005.csv$wage[!is.na(datind2005.csv$wage) & datind2005.csv$wage != 0]) # cannot do
mean (returns NA) if we don't omit missing data
```

```
## [1] 22443.02911846829
```

```
 sd(datind2005.csv$wage[!is.na(datind2005.csv$wage) & datind2005.csv$wage != 0]) # cannot do me
an (returns NA) if we don't omit missing data
```

```
## [1] 18076.708881794755
```

```
 D1 <- quantile(datind2005.csv$wage[!is.na(datind2005.csv$wage)& datind2005.csv$wage != 0],.1)
 D1
```

```
##  10%
## 4547
```

```
 D9 <- quantile(datind2005.csv$wage[!is.na(datind2005.csv$wage)& datind2005.csv$wage != 0],.9)
 D9 # is 32340.4
```

```
##                  90%
## 40452.500000000007
```

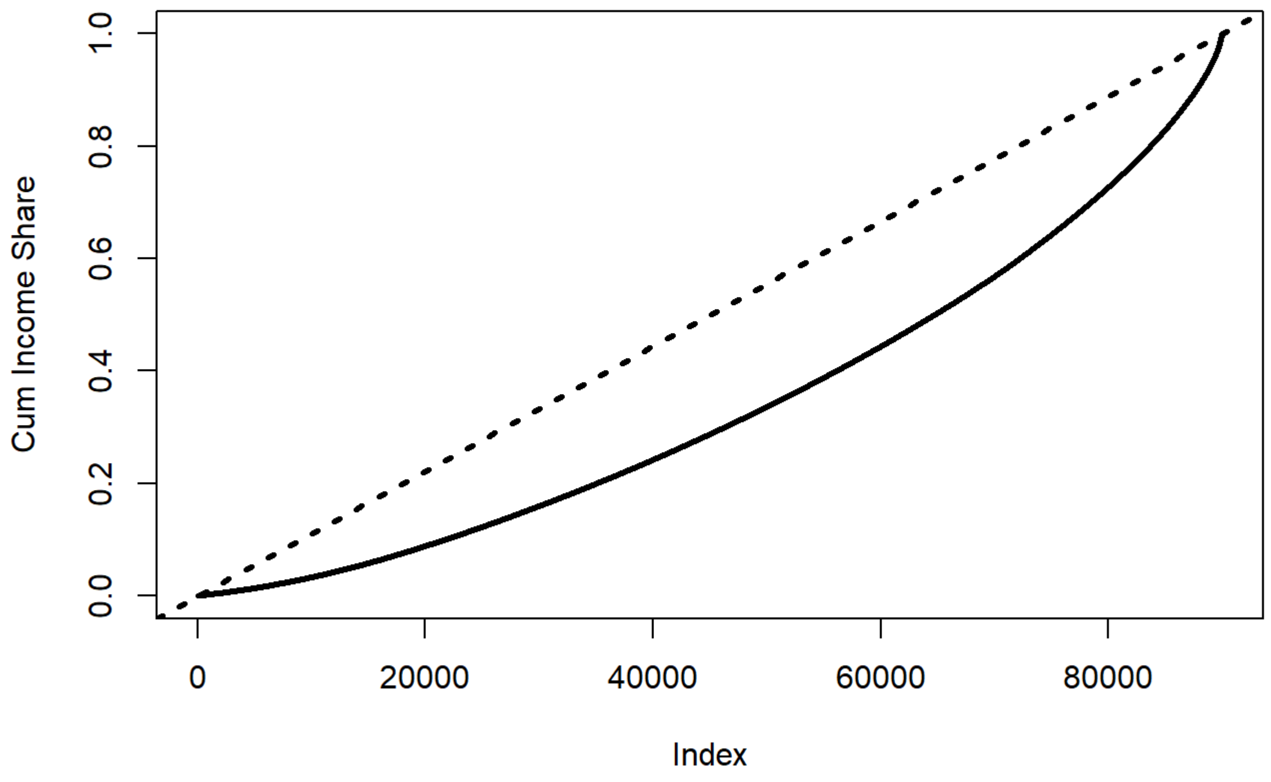
```
idr05 <- D9/D1
idr05[[1]]
```

```
## [1] 8.8965251814383119
```

```
#input into created function
inputData05 <- datind2005.csv$wage[!is.na(datind2005.csv$wage) & datind2005.csv$wage != 0]
GINI(inputData05)
```

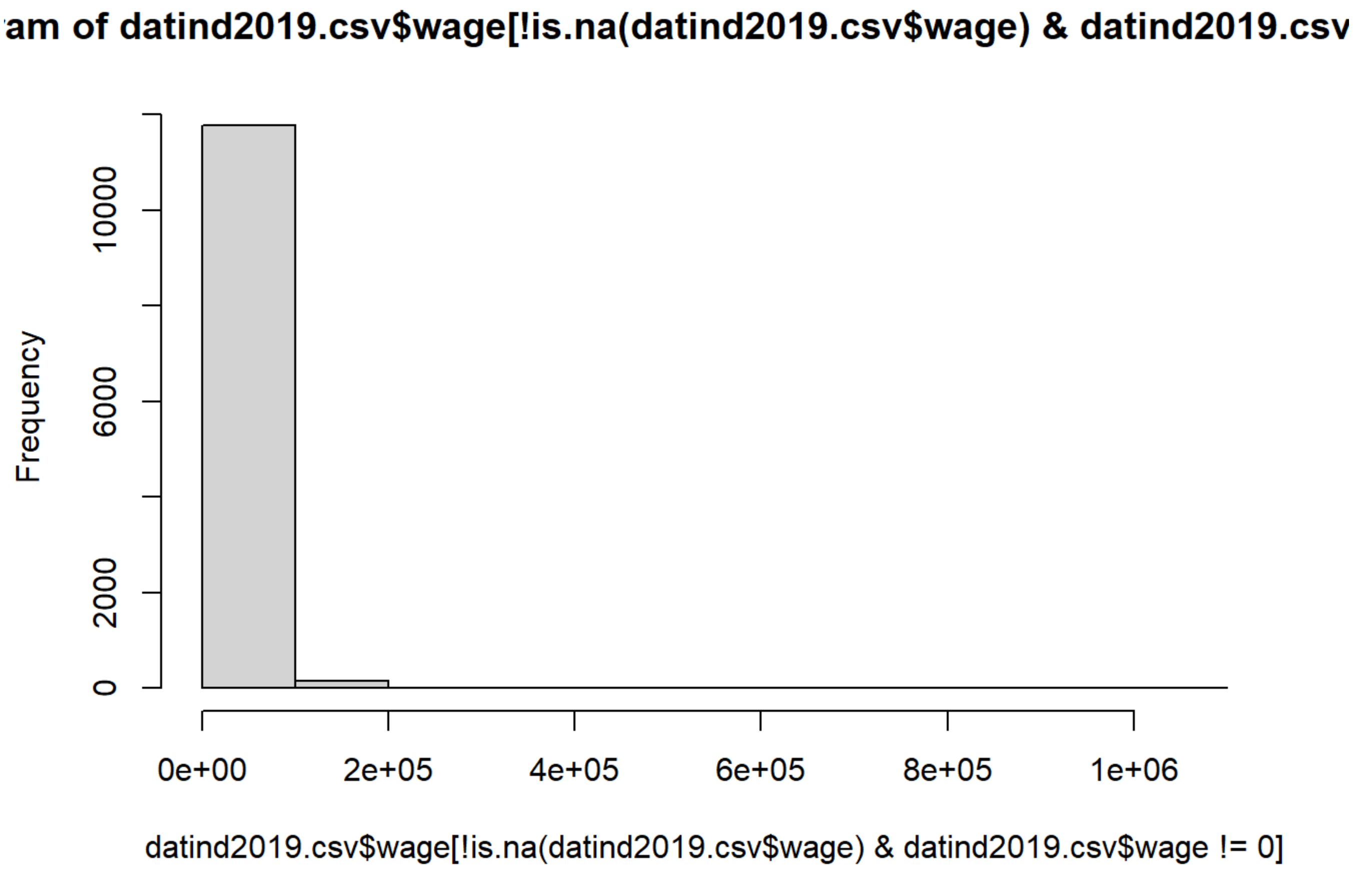
```
##
## Attaching package: 'pracma'
```

```
## The following object is masked from 'package:purrr':
##
##     cross
```



```
## [1] 0.3201291433600677
```

```
  #2019
hist(datind2019.csv$wage[!is.na(datind2019.csv$wage) & datind2019.csv$wage !=0])
```



```
mean(datind2019.csv$wage[!is.na(datind2019.csv$wage) & datind2019.csv$wage !=0])
```

```
## [1] 27578.839302189048
```

```
sd(datind2019.csv$wage[!is.na(datind2019.csv$wage) & datind2019.csv$wage !=0])
```

```
## [1] 25107.187195539096
```

```
  D1 <- quantile(datind2019.csv$wage[!is.na(datind2019.csv$wage) & datind2019.csv$wage !=0],.1
)
D1 # is 0
```

```
##                 10%
## 3634.0000000000005
```

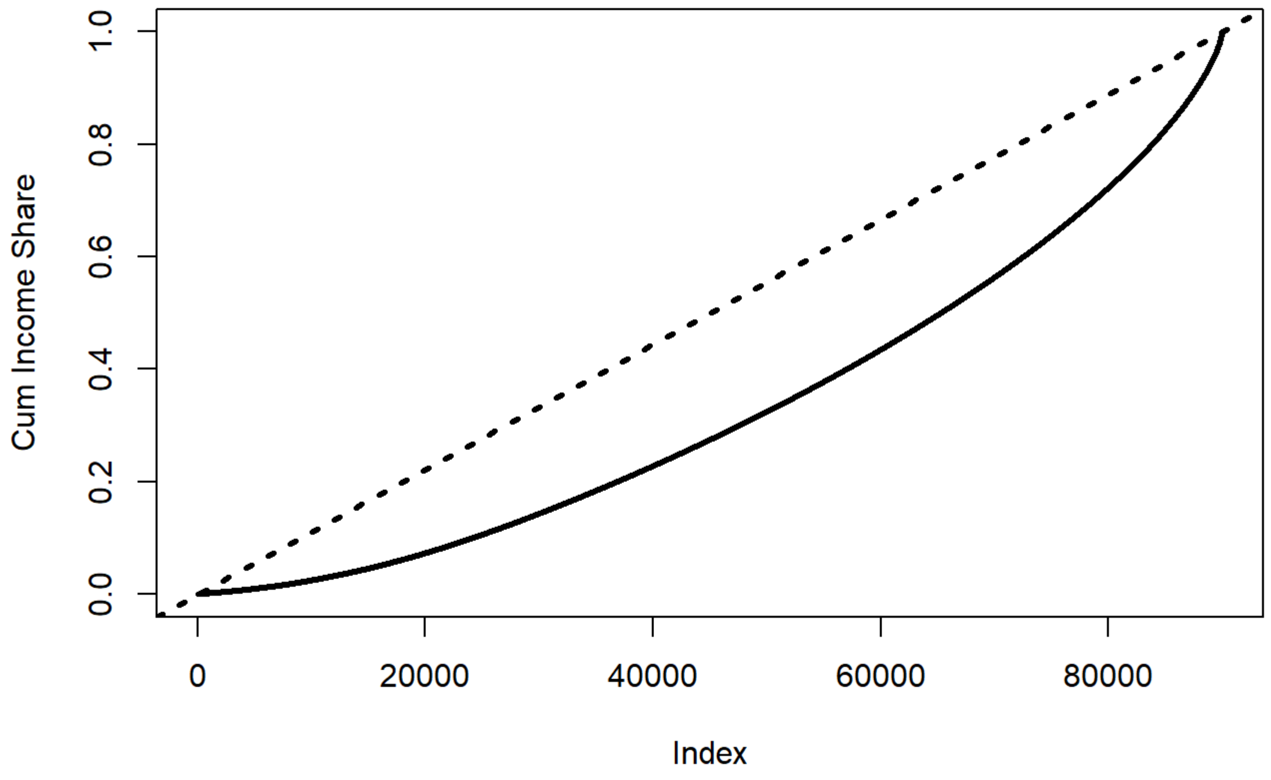
```
D9 <- quantile(datind2019.csv$wage[!is.na(datind2019.csv$wage) & datind2019.csv$wage !=0],.9)
D9 # is 40267
```

```
##                 90%
## 50375.600000000006
```

```
  idr19 <- D9/D1
  idr19[[1]]
```

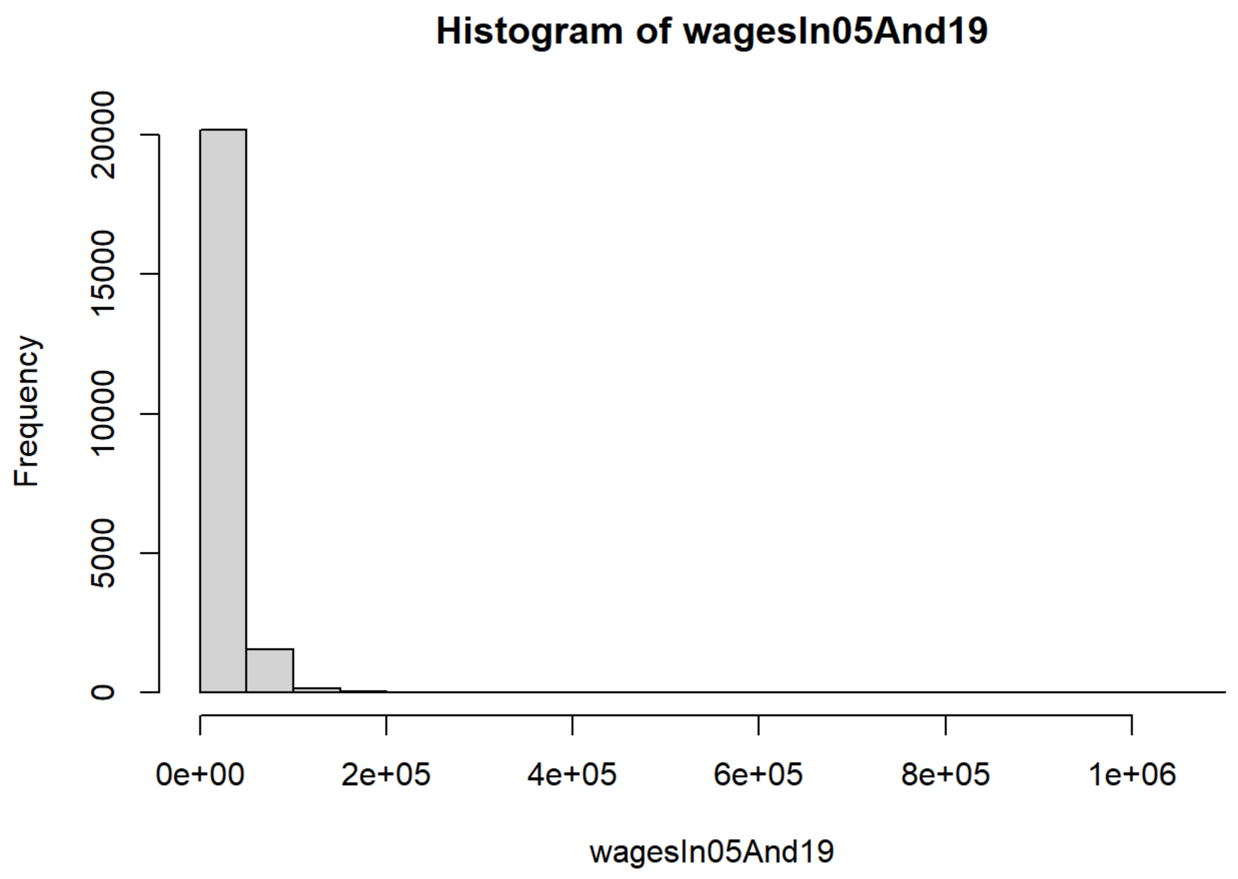
```
## [1] 13.862300495321959
```

```
  inputData19 <- datind2019.csv$wage[!is.na(datind2019.csv$wage) & datind2019.csv$wage !=0]
  GINI(inputData19)
```



```
## [1] 0.33903644704838326
```

```
  # together
  #combine wage data together into vector
  wagesIn05And19 <- c(datind2005.csv$wage[!is.na(datind2005.csv$wage) & datind2005.csv$wage !=0
],datind2019.csv$wage[!is.na(datind2019.csv$wage) & datind2019.csv$wage !=0])
  hist(wagesIn05And19)
```



```
#make sure no missing values
# wagesIn05And19 <- wagesIn05And19[!is.na(wagesIn05And19)]
# wagesIn05And19
mean(wagesIn05And19)
```

```
## [1] 25232.617967290786
```

```
sd(wagesIn05And19)
```

```
## [1] 22320.350540064475
```

```
D1 <- quantile(wagesIn05And19,.1)
D1
```

```
##  10%
## 3991
```

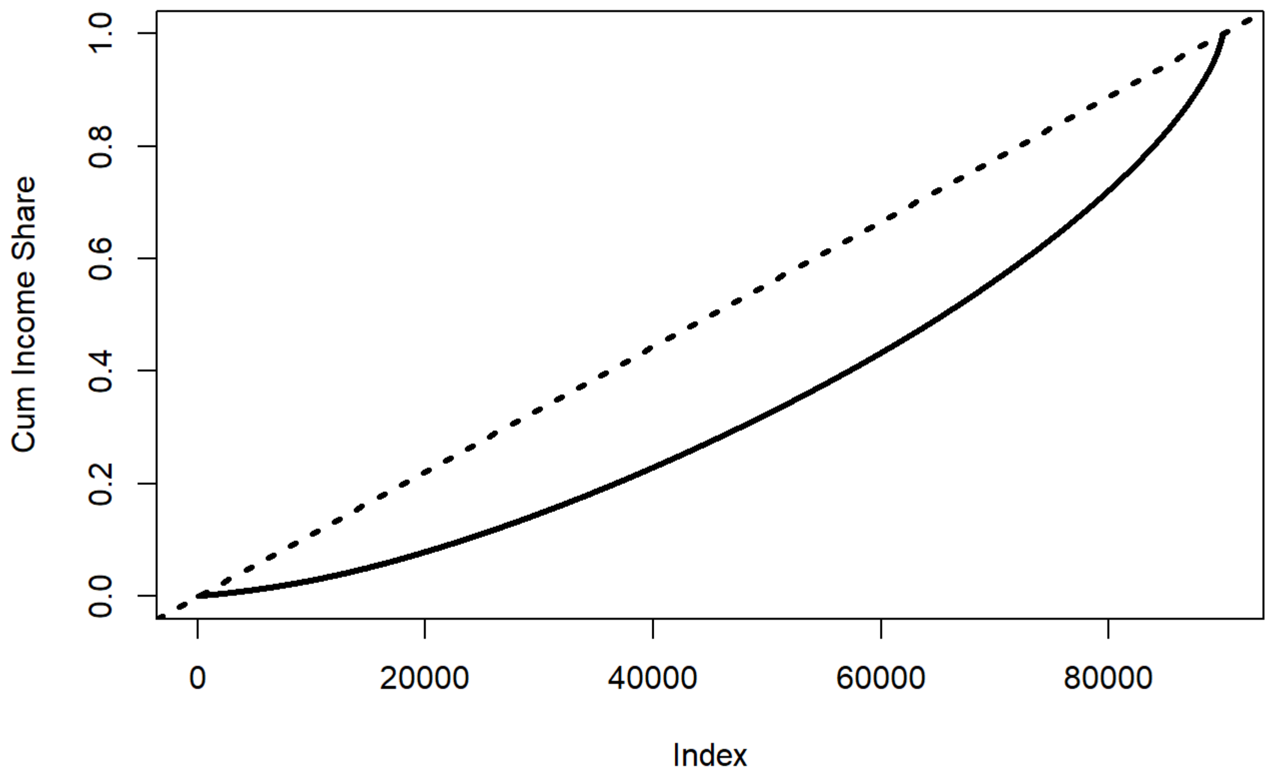
```
D9 <- quantile(wagesIn05And19,.9)
D9
```

```
##   90%
## 46219
```

```
  idr05And19 <- D9/D1
  idr05And19[[1]]
```

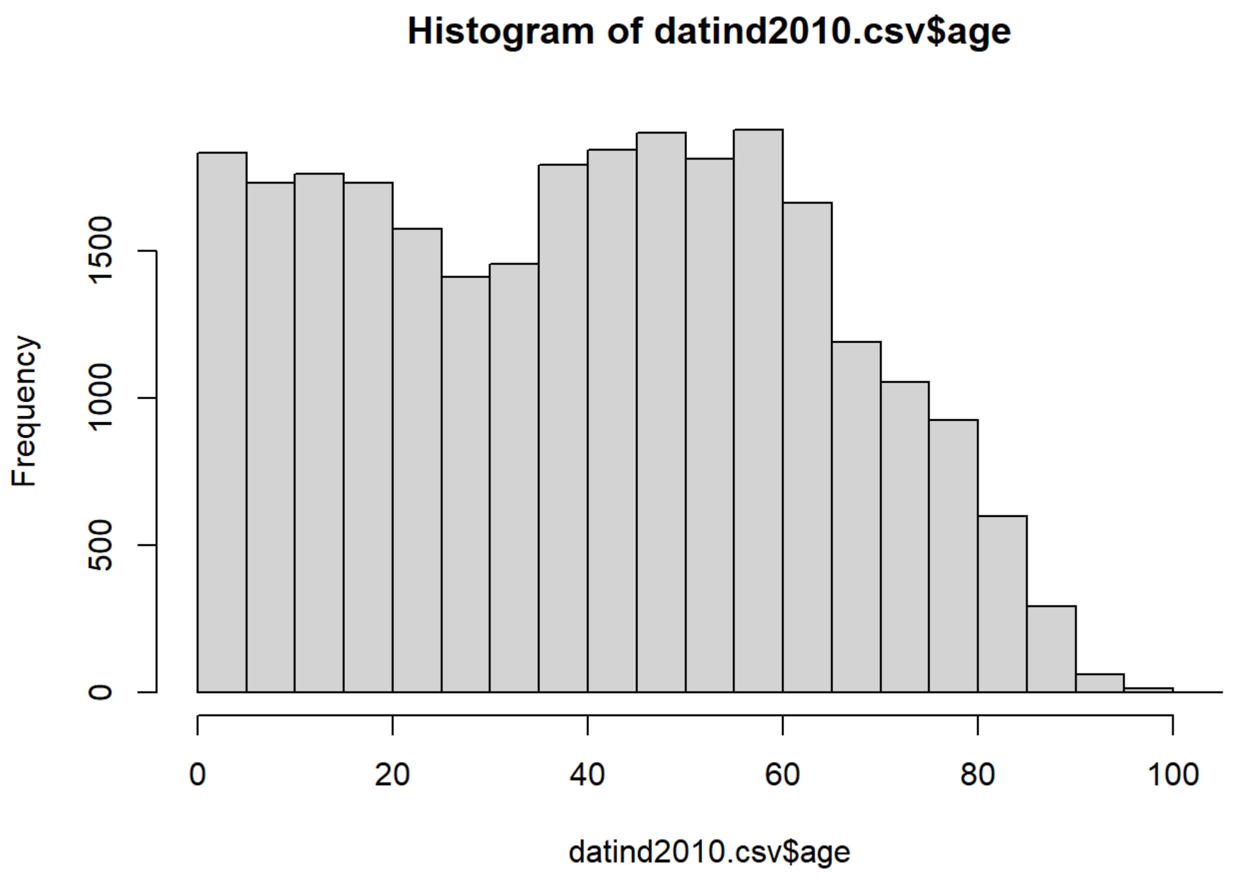
```
## [1] 11.580806815334503
```

```
  inputData05And19 <- wagesIn05And19
  GINI(inputData05And19)
```



```
## [1] 0.33725692054417167
```

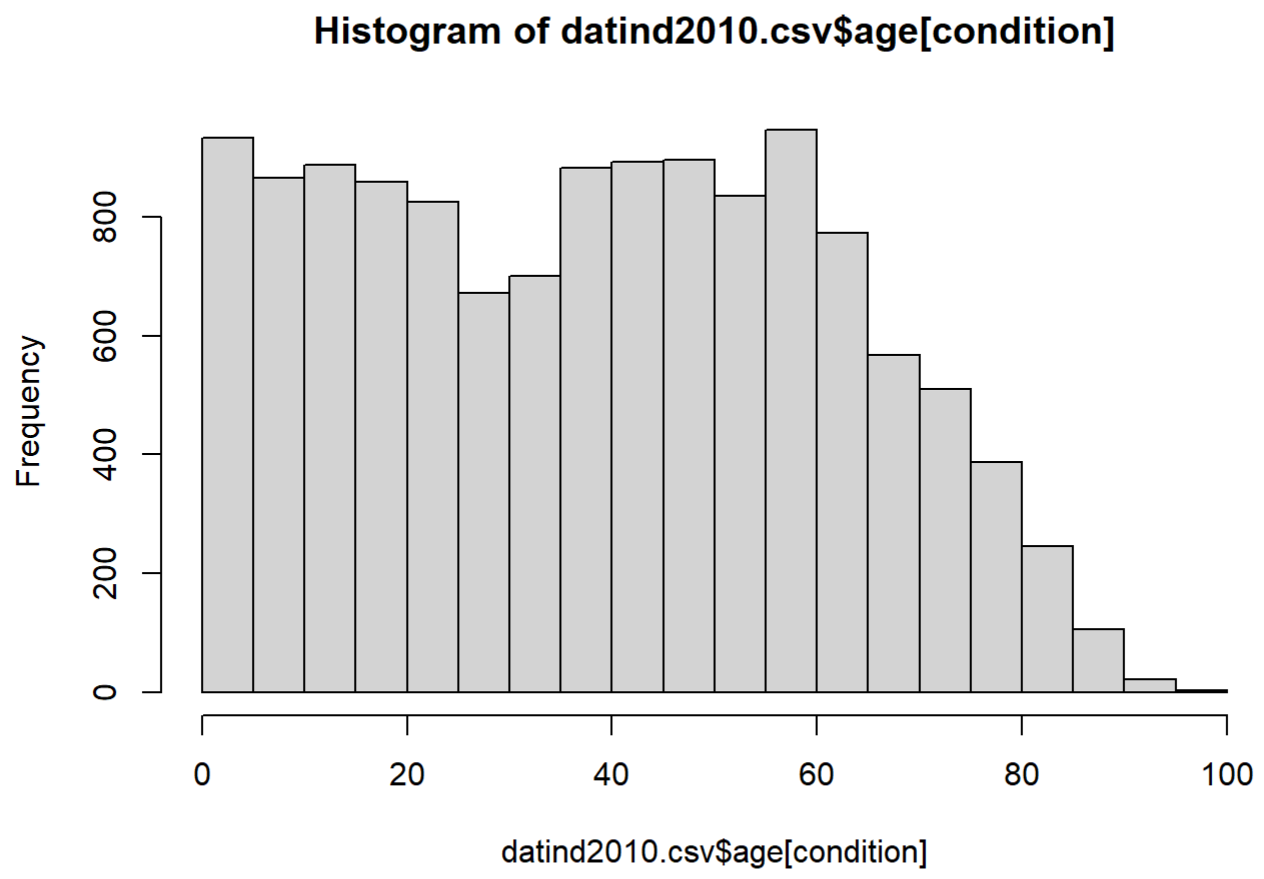
```
#g. distributio in age in 2010
hist(datind2010.csv$age)# general
```



```
summary(datind2010.csv$age)
```

```
##               Min.            1st Qu.             Median               Mean 
##   0.000000000000000  19.000000000000000  40.000000000000000  39.878934077117336 
##            3rd Qu.               Max. 
##  58.000000000000000 102.000000000000000
```

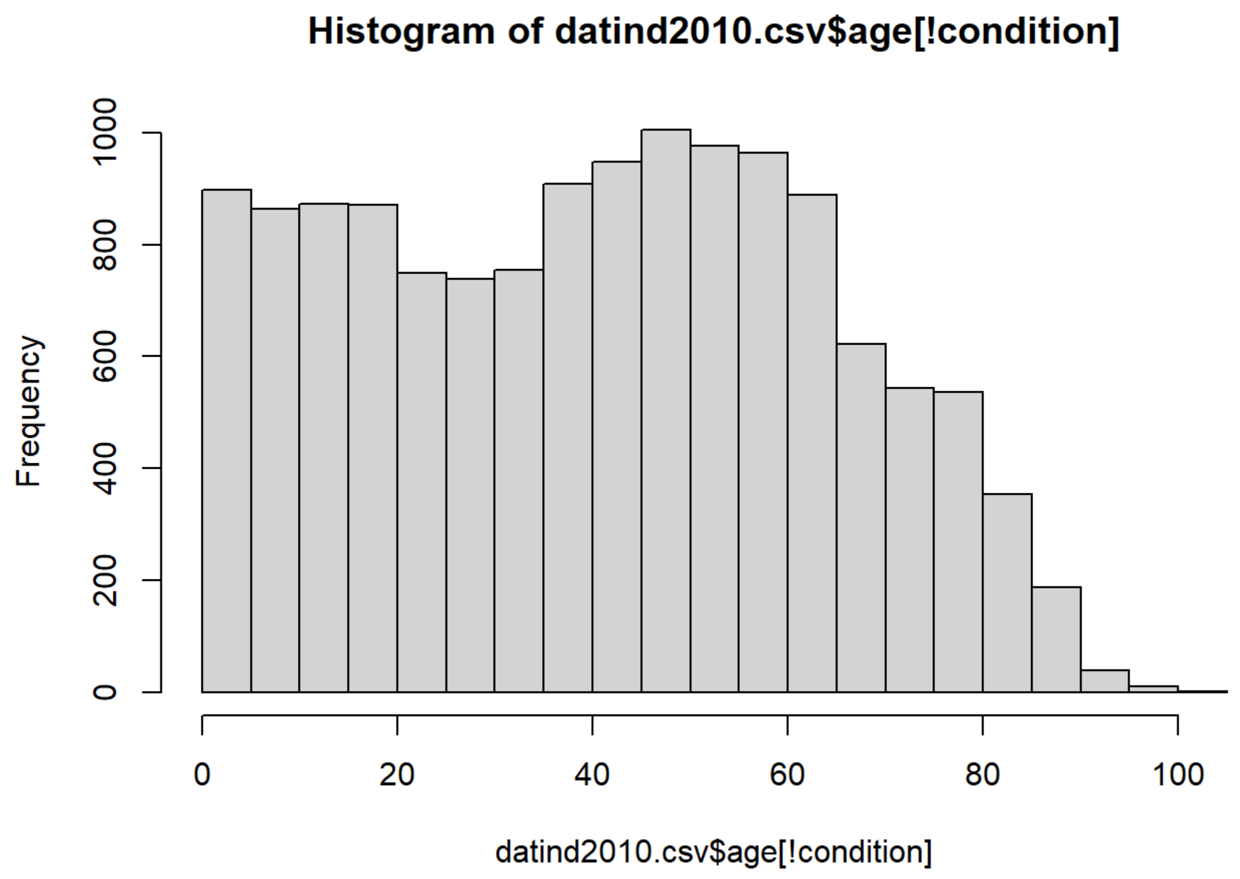
```
#definiting condition so that we look at just Males (to compare to females)
condition <-  datind2010.csv$gender == "Male"

#shape of distributions are similar but not identical so there is a difference between men and women
hist(datind2010.csv$age[condition])
```



```
summary(datind2010.csv$age[condition])
```

```
##               Min.           1st Qu.            Median              Mean 
##  0.000000000000000 19.000000000000000 39.000000000000000 38.873623369522768 
##            3rd Qu.              Max. 
## 57.000000000000000 96.000000000000000 
```

```
hist(datind2010.csv$age[!condition])
```



```
summary(datind2010.csv$age[!condition])
```

```
##               Min.           1st Qu.            Median              Mean 
##   0.000000000000000  20.000000000000000  42.000000000000000  40.816506410256409 
##            3rd Qu.              Max. 
##  59.000000000000000 102.000000000000000 
```

```
#h. number of individuals in Paris in 2011
#creating a vector of house id (assumed all unique even if idind match for some reason)
  hhNum <- datind2011.csv$idmen
  # hhNum
  # dathh2011.csv[998,8] == "Paris" # should be NA
  #count occurrences of Paris

  countInParis <- 0
  for (i in 1:length(hhNum)){
    #find matching household
    index <- match(hhNum[i],dathh2011.csv[,2])
    # if household location is not NA and is Paris, add to count
    if (!is.na(dathh2011.csv[index,8])){
      if(dathh2011.csv[index,8] == "Paris"){
        countInParis <- countInParis + 1
      }

    }
  }


  countInParis # 3514
```

```
## [1] 3514
```

```
    #pipeline group individuals by household number and summarize number of people in each house
hold
  anothereWay <- datind2011.csv %>% group_by(idmen) %>% summarize(count = n())

  # anothereWay

  counter <- 0
  for (i in 1:nrow(anothereWay)){
    #extract house id
    hhID <- anothereWay[[i,1]]
    #find indexed match in household data
    index <- match(hhID,dathh2011.csv[,2])
    #if household in Paris, add total number of household people to current value
    if (dathh2011.csv[index,8] == "Paris" & !is.na(dathh2011.csv[index,8])){
      counter <- counter + anothereWay[[i,2]]
    }
  }
  counter
```

```
## [1] 3514
```

```
  #same answer!
```

Exercise 2

```
#a. append all the individual datasets
#a read all the data (previously done) and combined them to form aggregate individual data
  #merge individual data
  indData <- datind2004.csv
  indData <- rbind(indData,datind2005.csv)
  indData <- rbind(indData,datind2006.csv)
  indData <- rbind(indData,datind2007.csv)
  indData <- rbind(indData,datind2008.csv)
  indData <- rbind(indData,datind2009.csv)
  indData <- rbind(indData,datind2010.csv)
  indData <- rbind(indData,datind2011.csv)
  indData <- rbind(indData,datind2012.csv)
  indData <- rbind(indData,datind2013.csv)
  indData <- rbind(indData,datind2014.csv)
  indData <- rbind(indData,datind2015.csv)
  indData <- rbind(indData,datind2016.csv)
  indData <- rbind(indData,datind2017.csv)
  indData <- rbind(indData,datind2018.csv)
  indData <- rbind(indData,datind2019.csv)

  length(unique(indData$idind)) #42868 with numerics. 100160 with characters
```

```
## [1] 100160
```

```
  length(unique(indData$idmen)) #41086 with numerics. 41086 with characters
```

```
## [1] 41086
```

```
#b. similar process as above but did it for household data
  #merge household data (by row)
  hhData <- dathh2004.csv
  hhData <- rbind(hhData,dathh2005.csv)
  hhData <- rbind(hhData,dathh2006.csv)
  hhData <- rbind(hhData,dathh2007.csv)
  hhData <- rbind(hhData,dathh2008.csv)
  hhData <- rbind(hhData,dathh2009.csv)
  hhData <- rbind(hhData,dathh2010.csv)
  hhData <- rbind(hhData,dathh2011.csv)
  hhData <- rbind(hhData,dathh2012.csv)
  hhData <- rbind(hhData,dathh2013.csv)
  hhData <- rbind(hhData,dathh2014.csv)
  hhData <- rbind(hhData,dathh2015.csv)
  hhData <- rbind(hhData,dathh2016.csv)
  hhData <- rbind(hhData,dathh2017.csv)
  hhData <- rbind(hhData,dathh2018.csv)
  hhData <- rbind(hhData,dathh2019.csv)

  length(unique(hhData$idmen)) #41084 with numerics and characters
```

```
## [1] 41084
```

```
#c. list all the variables present in both
 # colnames(indData)
  # colnames(hhData)
  intersect(  colnames(indData),
              colnames(hhData))
```

```
## [1] "X"     "idmen" "year"
```

```
  #matching variables: X, idmen, year
```

```
#d. merge the appended data sets
# creating subsets of data that have matching variables () idmen, and year) - omited X as it was
just a byproduct of reading in the data and would just mess up stuff

  aggData <- merge(hhData,indData,by = c("idmen","year"))
  nrow(aggData)
```

```
## [1] 413501
```

```
#e. Number of HH with more than 4 people
#group by HH and year and then find how many have more than 4 people associated with it
aggDatae <- aggData %>% group_by(idmen,year) %>% summarise(count = n())
```

```
## `summarise()` has grouped output by 'idmen'. You can override using the `.groups` argument.
```

```
# aggDatae
length(which(aggDatae[,"count"] > 4))
```

```
## [1] 12436
```

```
#f. Number of HH in which at least one member is UE
#pipeline group by idmen and year and see if ANY individual, in the household for a given year,
 is Unemployed by using %in%
  # will return TRUE if at least 1 person in Unemployed
  aggDataf <- aggData %>% group_by(idmen,year) %>% summarise(UE = "Unemployed" %in% empstat)
```

```
## `summarise()` has grouped output by 'idmen'. You can override using the `.groups` argument.
```

```
  # aggDataf

  length(which(aggDataf$UE == TRUE))
```

```
## [1] 17241
```

```
#g. Number of HH in which at least 2 have the same profession

# group by household and year and filter out professions that are missing or blank and then see
 if there are any Duplicates (anyDuplicated will return index of repeat if there is a duplicate
 and 0 otherwise)
aggDatag <- aggData %>% group_by(idmen,year) %>% filter(!is.na(profession), profession != "") %
>% summarise(sameProf = anyDuplicated(profession))
```

```
## `summarise()` has grouped output by 'idmen'. You can override using the `.groups` argument.
```

```
length(which(aggDatag$sameProf != 0))
```

```
## [1] 7586
```

```
#h. Num of individuals that are from Couple with Kids
# each row in aggData should be unique individual so just go through rows and check

countHHwithKids <- 0
  # each row should represent an individual in aggregate data so simply checking if that person
 is in a household "Couple, with Kids"
  for (i in 1:nrow(aggData)){
    if(aggData$mstatus[i] == "Couple, with Kids" & !is.na(aggData$mstatus[i])){
      countHHwithKids <- countHHwithKids + 1
    }
  }
  countHHwithKids
```

```
## [1] 209382
```

```
# quicker way
  length(which(aggData$mstatus == "Couple, with Kids"))
```

```
## [1] 209382
```

```
#i. number of individuals in Paris
#similar process as h - go through row of aggData and see if person in Paris. However, with ques
tion wording, I was unsure if it meant how many individuals were in Paris at some point in time
 in their survey responses so I did both

#if meant individual in a year - by row
  countInParis <- 0
  #each row should represent individual so simply check if that person is in "Paris"
  for (i in 1:nrow(aggData)){

    if (!is.na(aggData$location[i])){
      if (aggData$location[i] == "Paris"){
        countInParis <- countInParis + 1
      }
    }
  }
  countInParis
```

```
## [1] 51904
```

```
  #another way
  length(which(aggData$location == "Paris"))
```

```
## [1] 51904
```

```
  # #if mean unique individuals ever in Paris
  # uniqueIdind <- unique(aggData$idind)
  # aggDatai = aggData %>% group_by(idind) %>% summarize(fromParis = "Paris" %in% location)
  # length(which(aggDatai$fromParis == TRUE))
```

```
# j. idmen of largest household(s)
#group by household and year and measure ssize of group
  aggDataj <- aggData %>% group_by(idmen,year) %>% summarise(count = n())
```

```
## `summarise()` has grouped output by 'idmen'. You can override using the `.groups` argument.
```

```
  #find largest households in agiven year
  bigFam <- max(aggDataj[,3])
  #find index position of those
  which(aggDataj[,3] == bigFam) #there were 2
```

```
## [1]  69653 107200
```

```
  #find indmen and year
  idmen1Max <- aggDataj[[which(aggDataj[,3] == 14)[1],1]]
  idmen2Max <- aggDataj[[which(aggDataj[,3] == 14)[2],1]]
  idmen1MaxYear <- aggDataj[[which(aggDataj[,3] == 14)[1],2]]
  idmen2MaxYear <- aggDataj[[which(aggDataj[,3] == 14)[2],2]]

  #reporting them
  idmen1Max
```

```
## [1] "2207811124040100"
```

```
  idmen1MaxYear
```

```
## [1] 2007
```

```
  idmen2Max
```

```
## [1] "2510263102990100"
```

```
  idmen2MaxYear
```

```
## [1] 2010
```

```
#k. Number of households present in 2010 and 2011
nrow(dathh2010.csv) + nrow(dathh2011.csv)
```

```
## [1] 22408
```

```
# IF QUESTION MEANT HOW MANY HOUSEHOLDS WERE BOTH IN 2010 and 2011 (ie idmen in both 2010 and 20
11) but nothing about respondents

  aggDatak3 <- aggData %>% group_by(idmen) %>% summarise(present3 = ((2010 %in% year & 2011 %in%
year)))
  # aggDatak3


  length(which(aggDatak3$present3 == TRUE))
```

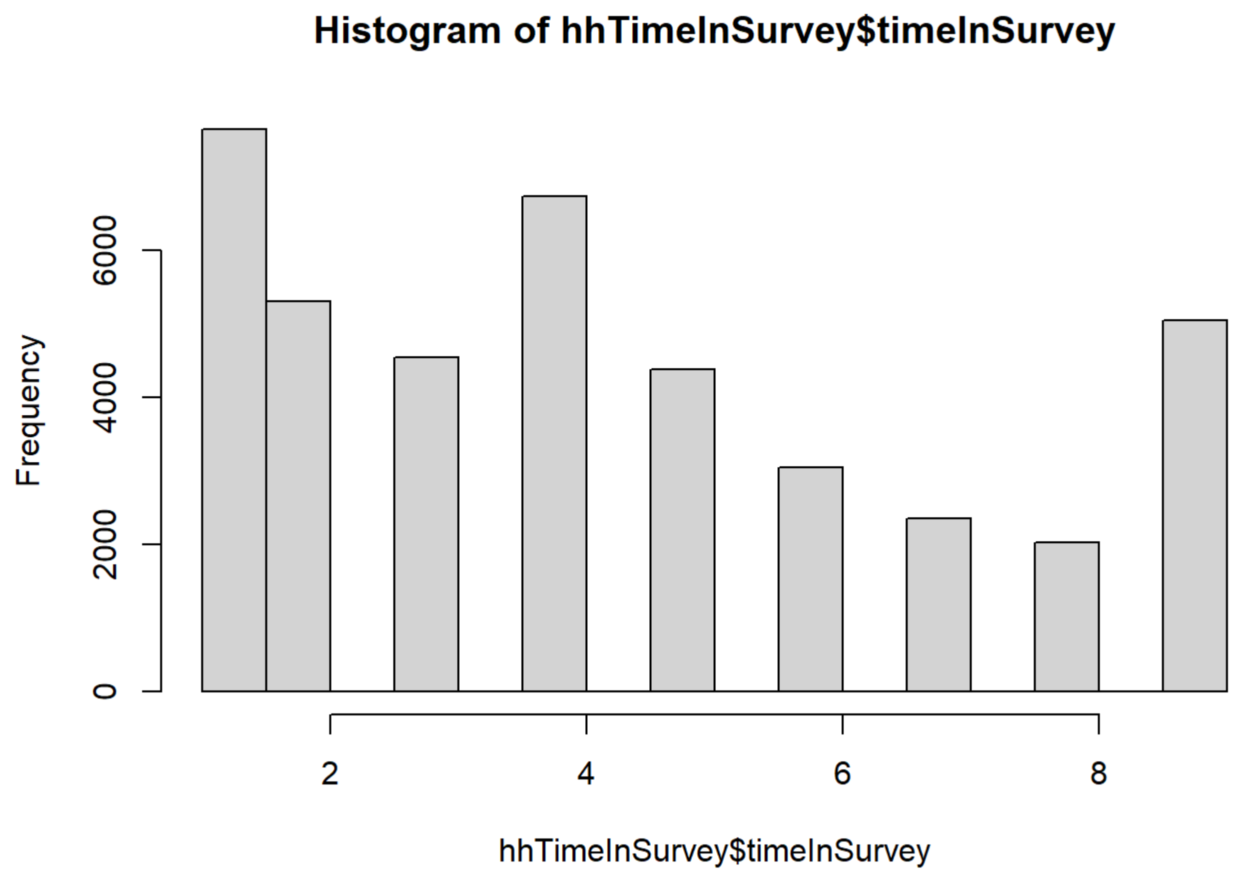
```
## [1] 8984
```

Exercise 3

```
#a. Find out each year household enters and exit the panel - report distribution of time spent f
or each household

# found total unique households
  uniqueIdmen <- unique(aggData$idmen) #getting unique household IDs
  length(uniqueIdmen) #41084 unique households
```

```
## [1] 41084
```

```
  # mutate data by household and put entry, exit

    hhEntryExit = aggData %>% group_by(idmen) %>% filter(!is.na(year)) %>% mutate(entryYear = mi
n(year),exitYear = max(year))

    # by household, find their time in survey by taking the differnence of their last and first
 year and adding 1
  hhTimeInSurvey = hhEntryExit %>% group_by(idmen) %>% summarise(timeInSurvey = max(year) - min
(year) + 1)

  # not getting unique households but individual level time spent if i don't use hhTimeInSurvey

  hist(hhTimeInSurvey$timeInSurvey)
```



```
  summary(hhTimeInSurvey$timeInSurvey)
```

```
##               Min.           1st Qu.            Median              Mean 
## 1.0000000000000000 2.0000000000000000 4.0000000000000000 4.3123357024632458 
##            3rd Qu.               Max. 
## 6.0000000000000000 9.0000000000000000
```

```
#
#   hhTimeInSurvey = aggData %>% group_by(idmen) %>% summarize(timeInSurvey = max(year) - min(ye
ar)+1)
#   hist(hhTimeInSurvey$timeInSurvey)
#   summary(hhTimeInSurvey$timeInSurvey)
#
```

```
#b. based on datent, check where houshold moved into current dwelling at time of year

# breaking into 2 parts since, for me, it seems more logical

#part 1 - looking at HH data and checking if each house had year == datent
  hhDataByYearb <- hhData %>% group_by(idmen,year) %>% mutate(movedAtYearOfSurvey = year == date
nt) # group by year and will be TRUE if year is same as datent

#reporting first 10 rows  of relevant variables
(hhDataByYearb[1:10,c("idmen","year","datent","movedAtYearOfSurvey")])
```

idmen <chr>	year <int>	datent <int>	movedAtYearOfSurvey <lgl>
1200010012930100	2004	2000	FALSE
1200010040580100	2004	2001	FALSE
1200010066630100	2004	2000	FALSE
1200010082450100	2004	1957	FALSE
1200010086440100	2004	2001	FALSE
1200010102990100	2004	1990	FALSE
1200010118450100	2004	2000	FALSE
1200020012930100	2004	1948	FALSE
1200020017390100	2004	1979	FALSE
1200020026420100	2004	1984	FALSE

1-10 of 10 rows

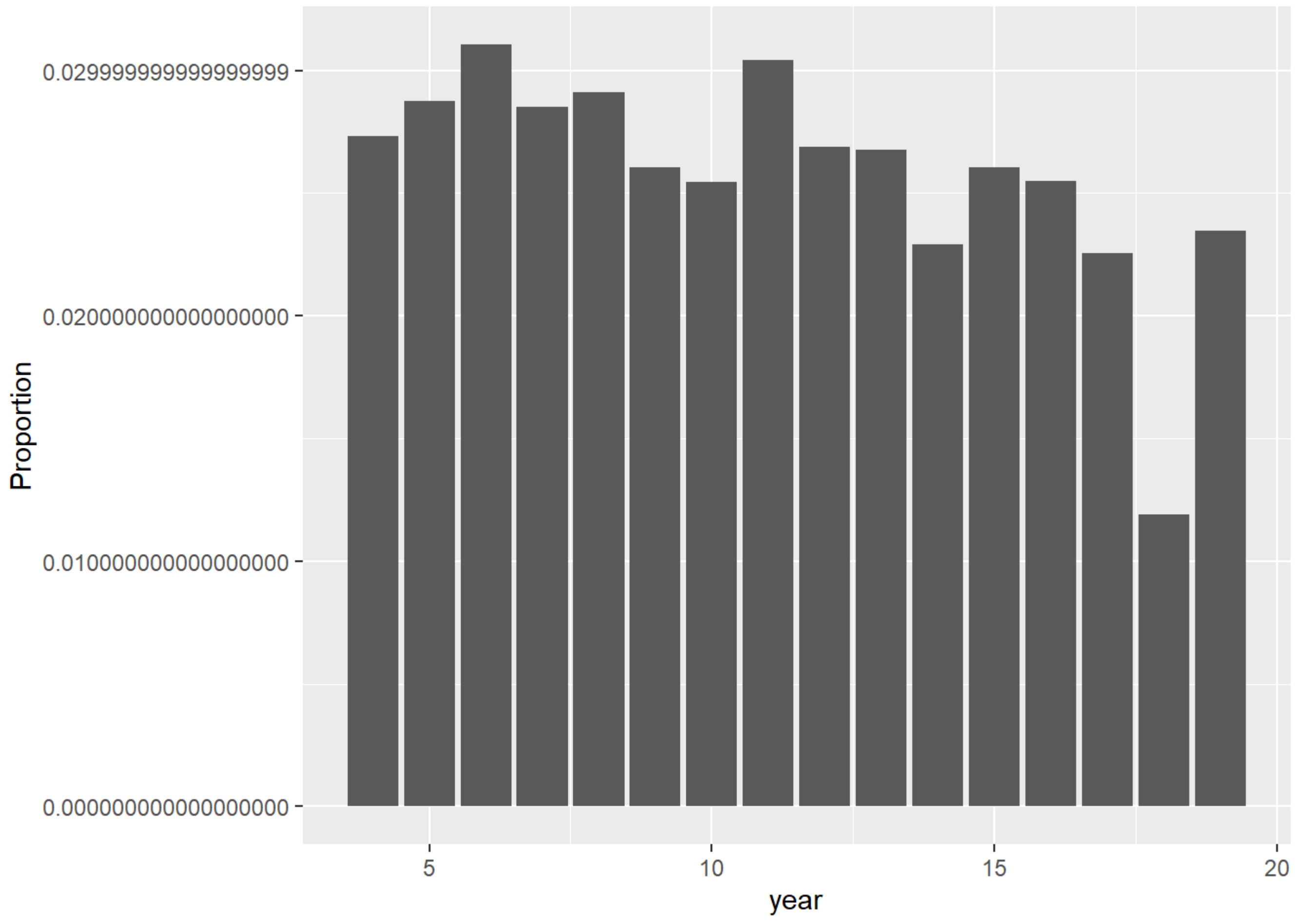
```
  nrow(hhDataByYearb)
```

```
## [1] 173851
```

```
  length(which(hhDataByYearb$movedAtYearOfSurvey == TRUE))
```

```
## [1] 4776
```

```
#part 2
# group by year and ID migration

method1ByYearMigration <- aggData %>% group_by(year) %>% filter(!is.na(year),!is.na(datent)) %>%
summarise(migration = length(which(year == datent)))

method1ByYearTotal <- aggData %>% group_by(year) %>% filter(!is.na(year),!is.na(datent)) %>% sum
marise(count = n())

#find proportions
shareofPopByYear <- method1ByYearMigration[,2]/method1ByYearTotal[,2]
# plotting
shareOfPopdf <- data.frame(c(04,05,06,07,08,09,10,11,12,13,14,15,16,17,18,19),shareofPopByYear)
  colnames(shareOfPopdf) <- c("year","Proportion")
  ggplot(shareOfPopdf, aes(x= year,y = Proportion))+
    geom_bar(stat="identity")
```



```
  shareOfPopM1 <- shareofPopByYear
```

```
#b. similar thing but using myear and move. myear available until 2014 and move available after.
Assuming move == 2 means moved (instead of 1)

# first part - whether households migrated that year or not
  hhDataByYearc <- hhData %>% group_by(idmen,year) %>% mutate(
    migration = if (year <= 2014){
    migration = year == myear # TRUE if year is equal to the move year
  }
  else if(year > 2014){
    migration = move == 2 # TRUE if they moved since last year (ie they moved this year)
  }
  )

(hhDataByYearc[1:10,c("idmen","year","myear","move","migration")])
```

idmen <chr>	year <int>	myear <int>	move <int>	migration <lgl>
1200010012930100	2004	2000	NA	FALSE
1200010040580100	2004	2001	NA	FALSE
1200010066630100	2004	2000	NA	FALSE
1200010082450100	2004	1957	NA	FALSE
1200010086440100	2004	2001	NA	FALSE
1200010102990100	2004	1990	NA	FALSE
1200010118450100	2004	2000	NA	FALSE
1200020012930100	2004	1988	NA	FALSE
1200020017390100	2004	1979	NA	FALSE
1200020026420100	2004	1981	NA	FALSE

1-10 of 10 rows

```
#part2 - same process as earlier, just different checks

method2ByYearMigration <- aggData %>% group_by(year) %>% summarise(migration = length(which(
    if (year <= 2014){
    migration = year == myear # TRUE if year is equal to the move year
  }
  else if(year > 2014){
    migration = move == 2 # TRUE if they moved since last year (ie they moved this year)
  })))
```

```
## Warning in if (year <= 2014) {: the condition has length > 1 and only the first
## element will be used

## Warning in if (year <= 2014) {: the condition has length > 1 and only the first
## element will be used

## Warning in if (year <= 2014) {: the condition has length > 1 and only the first
## element will be used

## Warning in if (year <= 2014) {: the condition has length > 1 and only the first
## element will be used

## Warning in if (year <= 2014) {: the condition has length > 1 and only the first
## element will be used

## Warning in if (year <= 2014) {: the condition has length > 1 and only the first
## element will be used

## Warning in if (year <= 2014) {: the condition has length > 1 and only the first
## element will be used

## Warning in if (year <= 2014) {: the condition has length > 1 and only the first
## element will be used

## Warning in if (year <= 2014) {: the condition has length > 1 and only the first
## element will be used

## Warning in if (year <= 2014) {: the condition has length > 1 and only the first
## element will be used

## Warning in if (year <= 2014) {: the condition has length > 1 and only the first
## element will be used

## Warning in if (year <= 2014) {: the condition has length > 1 and only the first
## element will be used
```

```
## Warning in if (year > 2014) {: the condition has length > 1 and only the first
## element will be used
```

```
## Warning in if (year <= 2014) {: the condition has length > 1 and only the first
## element will be used
```

```
## Warning in if (year > 2014) {: the condition has length > 1 and only the first
## element will be used
```

```
## Warning in if (year <= 2014) {: the condition has length > 1 and only the first
## element will be used
## Warning in if (year > 2014) {: the condition has length > 1 and only the first
## element will be used
```

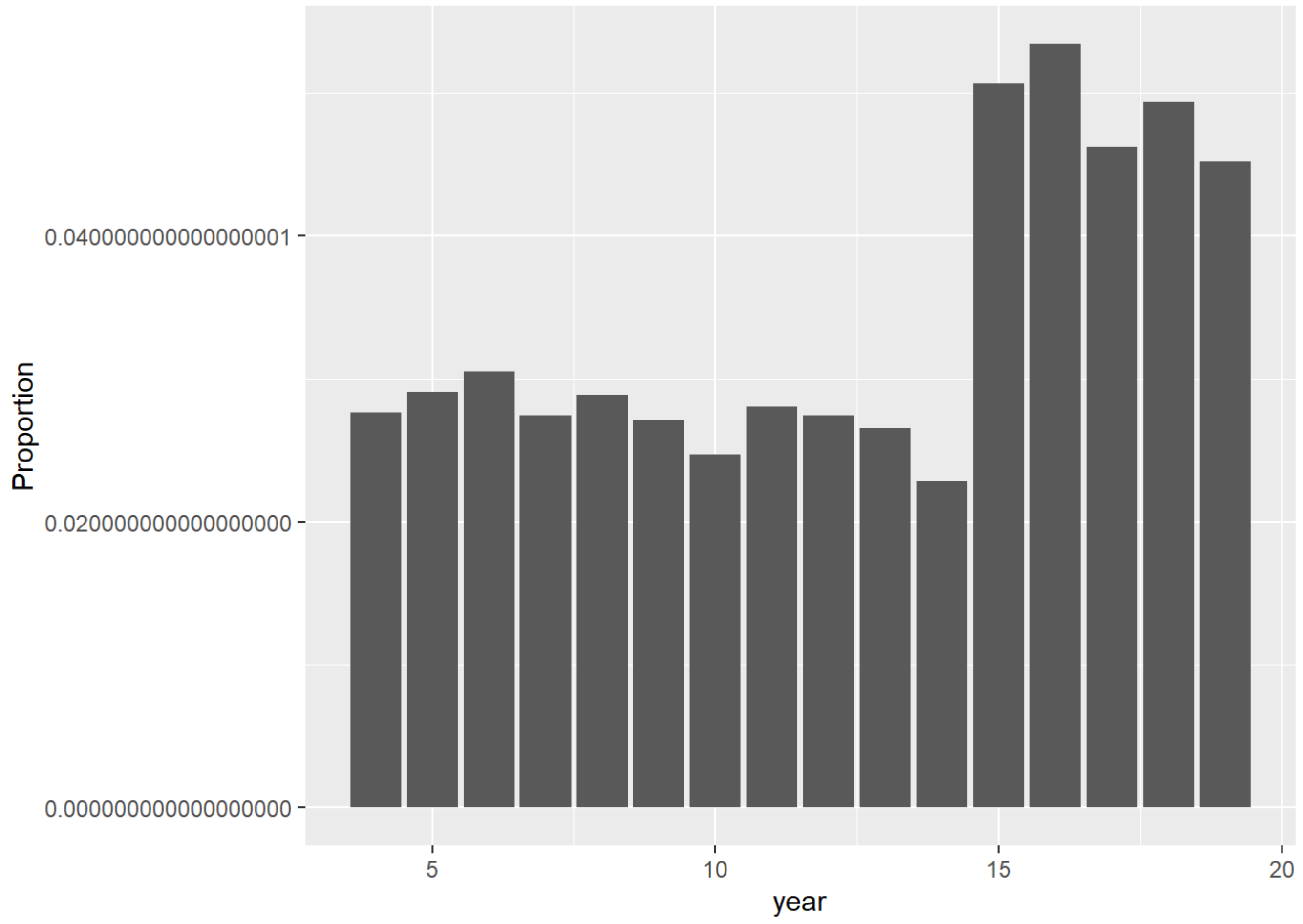
```
## Warning in if (year <= 2014) {: the condition has length > 1 and only the first
## element will be used
```

```
## Warning in if (year > 2014) {: the condition has length > 1 and only the first
## element will be used
```

```
## Warning in if (year <= 2014) {: the condition has length > 1 and only the first
## element will be used
```

```
## Warning in if (year > 2014) {: the condition has length > 1 and only the first
## element will be used
```

```
  method2ByYearTotal <- aggData %>% group_by(year) %>% summarise(count = n())

  shareofPopByYear <- method2ByYearMigration[,2]/method2ByYearTotal[,2]
  shareOfPopM2 <- shareofPopByYear

  shareOfPopdf <- data.frame(c(04,05,06,07,08,09,10,11,12,13,14,15,16,17,18,19),shareofPopByYea
r)
  colnames(shareOfPopdf) <- c("year","Proportion")
  ggplot(shareOfPopdf, aes(x= year,y = Proportion))+
    geom_bar(stat="identity")
```



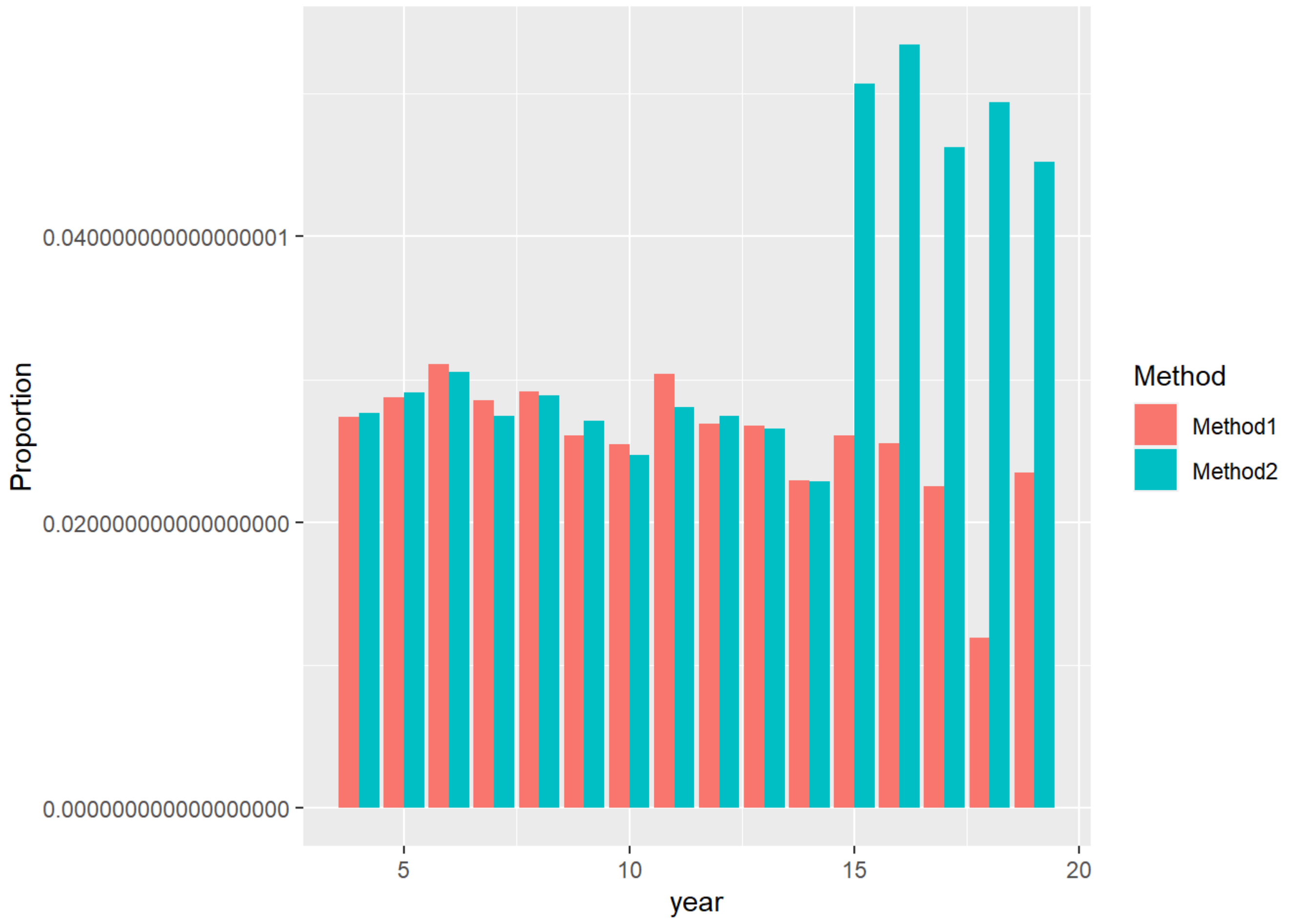
```
nrow(hhDataByYearc)
```

```
## [1] 173851
```

```
length(which(hhDataByYearc$migration == TRUE))
```

```
## [1] 6071
```

```
#d. merge the data


  method1 <- rep("Method1",16)
  method2 <- rep("Method2",16)
  method <- c(method1,method2)
  shareOfProportion <- rbind(shareOfPopM1,shareOfPopM2)

  #create data frame
  combinedData <- data.frame(c(04,05,06,07,08,09,10,11,12,13,14,15,16,17,18,19),method,shareOfPr
oportion)
  # View(combinedData)
  colnames(combinedData) <- c("year","Method","Proportion")

  ggplot(combinedData, aes(x=year, y = Proportion, fill = Method)) +
    geom_bar(stat = "identity",position = "dodge")
```



```
  length(which(is.na(aggData[,2]))) #none missing
```

```
## [1] 0
```

```
  length(which(is.na(aggData[,4])))
```

```
## [1] 245
```

```
  #245 missing

    in04 <- which(aggData[,2] == 2004)

  in05 <- which(aggData[,2] == 2005)

  in06 <- which(aggData[,2] == 2006)

  in07 <- which(aggData[,2] == 2007)

  in08 <- which(aggData[,2] == 2008)

  in09 <- which(aggData[,2] == 2009)

  in10 <- which(aggData[,2] == 2010)

  in11 <- which(aggData[,2] == 2011)

  in12 <- which(aggData[,2] == 2012)

  in13 <- which(aggData[,2] == 2013)

  in14 <- which(aggData[,2] == 2014)

  in15 <- which(aggData[,2] == 2015)

  in16 <- which(aggData[,2] == 2016)

  in17 <- which(aggData[,2] == 2017)

  in18 <- which(aggData[,2] == 2018)

  in19 <- which(aggData[,2] == 2019)
  # below only work if you uncomment inXY from above
  until14 <- c(in04,in05,in06,in07,in08,in09,in10,in11,in12,in13,in14)
  after14 <- c(in15,in16,in17,in18,in19)
  length(which(is.na(aggData[until14,5])))
```

```
## [1] 6446
```

```
  #6446 missing
  length(which(is.na(aggData[after14,7])))
```

```
## [1] 25250
```

```
  #25250 missing
```

```
#e. Find out how many households had at least one family member change profession/empstat

# first issue is determining which criteria to use to do this - question said migrate and that w
as associated with method2 as opposed to method1(which used moved). That is why I chose to use t
hat method. If time permits, I will also try to do it with method1 as I have previously said it
 is better (more data)

#using method2
method2Migration <- hhData %>% group_by(idmen,year) %>% summarise(
    migration = if (year <= 2014){
    migration = year == myear # TRUE if year is equal to the move year
  }
  else if(year > 2014){
    migration = move == 2}) # TRUE if they moved since last year (ie they moved this year)
```

```
## `summarise()` has grouped output by 'idmen'. You can override using the `.groups` argument.
```

```
#individuals in houses that have migrated (according to method2)



newAggData <- merge(aggData,method2Migration, by = c("idmen","year"))
#add column of 0s to newAggData to track job movements
  Change <- rep(0,nrow(newAggData))
  newAggData <- cbind(newAggData,Change) #add column of zeros at the end which will then be used
for tracking job changes
migratedHouseData <- newAggData[ which(newAggData$migration == TRUE),]

  # extract unique people
  uniqueIdind <- unique(migratedHouseData$idind)
  # length(uniqueIdind)

  #go through list
  for (i in 1:length(uniqueIdind)){
    ID <- uniqueIdind[i]
    #find indices corresponding to them
    index <- which(migratedHouseData[,"idind"] == ID)

    #check if empstat or profesison changed

    for (j in 1:length(index)){
      if (j > 1){

        #get index for years
        lastYearIndex <- index[j-1]
        thisYearIndex <- index[j]


        if(!is.na(migratedHouseData[lastYearIndex,"profession"]) &
           !is.na(migratedHouseData[thisYearIndex,"profession"])){
    #checked if not missing professions between this year and last year
          #check if not blank both years and if they are not the same (ie change occurred)
          if((migratedHouseData[lastYearIndex,"profession"] != "" &
              migratedHouseData[thisYearIndex,"profession"] != "") &
             (migratedHouseData[lastYearIndex,"profession"] !=
              migratedHouseData[thisYearIndex,"profession"])){

            migratedHouseData[thisYearIndex,"Change"] <- 1
          }
        }



          # check for change in empstat if profession doesnt change

          if (!is.na(migratedHouseData[lastYearIndex,"empstat"]) &
              !is.na(migratedHouseData[thisYearIndex,"empstat"])){
          if (migratedHouseData[lastYearIndex,"empstat"] != migratedHouseData[thisYearIndex,"emp
stat"]){
            migratedHouseData[thisYearIndex,"Change"] <- 1
          }
        }

      }

    }
  }
  # group and sum up changes (no changes in profession/employment would equal 0)

  # sum will be 0 if no change in that year
  jobChangeAndMove <- migratedHouseData %>% group_by(idmen,year) %>% summarise(jobChange = sum(C
hange) )
```

```
## `summarise()` has grouped output by 'idmen'. You can override using the `.groups` argument.
```

```
  length(which(jobChangeAndMove[,3] > 0))
```

```
## [1] 518
```

```
  jobChangeByYear2 <-  jobChangeAndMove %>% group_by(year) %>%
  summarise(changeInProfOrEmp =  length(which(jobChange!= 0)))

  jobChangeByYear2
```

year <int>	changeInProfOrEmp <int>
2004	0
2005	4
2006	12
2007	12
2008	28
2009	27
2010	21
2011	28
2012	34
2013	38

1-10 of 16 rows Previous **1** 2 Next

```
  length(which(jobChangeAndMove[,3] > 0))
```

```
## [1] 518
```

```
  #####################
  #using method1
method1Migration <- hhData %>% group_by(idmen,year) %>% summarise(
    migration = year == datent) # TRUE if they moved since last year (ie they moved this year)
```

```
## `summarise()` has grouped output by 'idmen'. You can override using the `.groups` argument.
```

```r
#individuals in houses that have migrated (according to method2)




newAggData <- merge(aggData,method1Migration, by = c("idmen","year"))
#add column of 0s to newAggData to track job movements
  Change <- rep(0,nrow(newAggData))

  newAggData <- cbind(newAggData,Change) #add column of zeros at the end which will then be used
for tracking job changes
migratedHouseData <- newAggData[ which(newAggData$migration == TRUE),]

# extract unique people
  uniqueIdind <- unique(migratedHouseData$idind)
  # length(uniqueIdind)




#extract unique people
  uniqueIdind <- unique(migratedHouseData$idind)
  # length(uniqueIdind)

  #go through list
  for (i in 1:length(uniqueIdind)){
    ID <- uniqueIdind[i]
    #find indices corresponding to them
    index <- which(migratedHouseData[,"idind"] == ID)

    #check if empstat or profesison changed

    for (j in 1:length(index)){
      if (j > 1){

        #get index for years
        lastYearIndex <- index[j-1]
        thisYearIndex <- index[j]

        if(!is.na(migratedHouseData[lastYearIndex,"profession"]) &
           !is.na(migratedHouseData[thisYearIndex,"profession"])){

              #checked if not missing professions between this year and last year
          #check if not blank both years and if they are not the same (ie change occurred)

          if((migratedHouseData[lastYearIndex,"profession"] != "" &
              migratedHouseData[thisYearIndex,"profession"] != "") &
             (migratedHouseData[lastYearIndex,"profession"] !=
              migratedHouseData[thisYearIndex,"profession"])){

            migratedHouseData[thisYearIndex,"Change"] <- 1
          }
        }
```

```
          # check for change in empstat if profession doesnt change

          if (!is.na(migratedHouseData[lastYearIndex,"empstat"]) &
              !is.na(migratedHouseData[thisYearIndex,"empstat"])){
          if (migratedHouseData[lastYearIndex,"empstat"] != migratedHouseData[thisYearIndex,"emp
stat"]){
            migratedHouseData[thisYearIndex,"Change"] <- 1
          }
        }

      }

    }
  }
  # group and sum up changes (no changes in profession/employment would equal 0)

  # sum will be 0 if no change in that year
  jobChangeAndMove <- migratedHouseData %>% group_by(idmen,year) %>% summarise(jobChange = sum(C
hange) )
```

```
## `summarise()` has grouped output by 'idmen'. You can override using the `.groups` argument.
```

```
  length(which(jobChangeAndMove[,3] > 0))
```

```
## [1] 312
```

```
    jobChangeByYear1 <-  jobChangeAndMove %>% group_by(year) %>%
    summarise(changeInProfOrEmp =  length(which(jobChange!= 0)))
  jobChangeByYear1
```

year <int>	changeInProfOrEmp <int>
2004	1
2005	5
2006	14
2007	13
2008	24
2009	25
2010	22
2011	30
2012	34

year <int>	changeInProfOrEmp <int>
2013	40

1-10 of 16 rows Previous **1** 2 Next

```
  length(which(jobChangeAndMove[,3] > 0))
```

```
## [1] 312
```

```
###########################

  #using either
methodMergedMigration <- hhData %>% group_by(idmen,year) %>% summarise(
    migration = year == datent |
    if (year <= 2014){
    migration = year == myear # TRUE if year is equal to the move year
  }
  else if(year > 2014){
    migration = move == 2}
  ) # TRUE if they moved since last year (ie they moved this year)
```

```
## `summarise()` has grouped output by 'idmen'. You can override using the `.groups` argument.
```

```
#individuals in houses that have migrated (according to method2)




newAggData <- merge(aggData,methodMergedMigration, by = c("idmen","year"))
#add column of 0s to newAggData to track job movements
  Change <- rep(0,nrow(newAggData))
  newAggData <- cbind(newAggData,Change) #add column of zeros at the end which will then be used
for tracking job changes
migratedHouseData <- newAggData[ which(newAggData$migration == TRUE),]

#extract unique people
  uniqueIdind <- unique(migratedHouseData$idind)
  # length(uniqueIdind)

  #go through list
  for (i in 1:length(uniqueIdind)){
    ID <- uniqueIdind[i]
    #find indices corresponding to them
    index <- which(migratedHouseData[,"idind"] == ID)

    #check if empstat or profesison changed

    for (j in 1:length(index)){
      if (j > 1){

        #get index for years
        lastYearIndex <- index[j-1]
        thisYearIndex <- index[j]


        if(!is.na(migratedHouseData[lastYearIndex,"profession"]) &
           !is.na(migratedHouseData[thisYearIndex,"profession"])){

              #checked if not missing professions between this year and last year
          #check if not blank both years and if they are not the same (ie change occurred)

            if((migratedHouseData[lastYearIndex,"profession"] != "" &
              migratedHouseData[thisYearIndex,"profession"] != "") &
             (migratedHouseData[lastYearIndex,"profession"] !=
              migratedHouseData[thisYearIndex,"profession"])){

            migratedHouseData[thisYearIndex,"Change"] <- 1
          }
        }


          # check for change in empstat if profession doesnt change

        if (!is.na(migratedHouseData[lastYearIndex,"empstat"]) &
              !is.na(migratedHouseData[thisYearIndex,"empstat"])){
```

```
          if (migratedHouseData[lastYearIndex,"empstat"] != migratedHouseData[thisYearIndex,"emp
stat"]){
            migratedHouseData[thisYearIndex,"Change"] <- 1
          }
        }

      }

    }
  }

  # group and sum up changes (no changes in profession/employment would equal 0)

  # sum will be 0 if no change in that year
  jobChangeAndMove <- migratedHouseData %>% group_by(idmen,year) %>% summarise(jobChange = sum(C
hange))
```

```
## `summarise()` has grouped output by 'idmen'. You can override using the `.groups` argument.
```

```
  # jobChangeAndMove

  jobChangeByYearBoth <-  jobChangeAndMove %>% group_by(year) %>%
    summarise(changeInProfOrEmp =  length(which(jobChange!= 0)))
  jobChangeByYearBoth
```

year <int>	changeInProfOrEmp <int>
2004	1
2005	7
2006	18
2007	15
2008	29
2009	29
2010	22
2011	35
2012	41
2013	45

1-10 of 16 rows Previous **1** 2 Next

```
  length(which(jobChangeAndMove[,3] > 0))
```

```
## [1] 614
```

```
  View(jobChangeByYear2)
  View(jobChangeByYear1)
  View(jobChangeByYearBoth)
```

Exercise 4

```
#4
minMaxAttrition <- aggData %>% group_by(idind) %>% summarise(entry = min(year),exit = max(year))

(minMaxAttrition)
```

idind <chr>	**entry** <int>	**exit** <int>
1120001001293010001	2004	2004
1120001004058010001	2004	2005
1120001004058010002	2004	2005
1120001006663010001	2004	2005
1120001006663010002	2004	2005
1120001008245010001	2004	2005
1120001008644010001	2004	2005
1120001008644010002	2004	2005
1120001010299010001	2004	2005
1120001010299010002	2004	2005

1-10 of 10,000 rows Previous **1** 2 3 4 5 6 ... 1000 Next

```
# now we have entry and exit data for each individual

totalInd <- c()
attritionPerYear <- c()
for (i in 2004:2019){
  year <- i
  count <- 0
  attrition <- 0
  # count how many people in the survey at that time (ie this year is within their entry exit ye
ars)
  count <- length(which(minMaxAttrition$entry <= i & i <= minMaxAttrition$exit))
#find how many people are leaving (ie this year is their exit year)
    attrition <- length(which(minMaxAttrition$exit == i))

  #attach to vectors so we get these values for each year
  totalInd <- c(totalInd,count)
  attritionPerYear <- c(attritionPerYear,attrition)
}

# totalInd
# attritionPerYear
propAtrrition <- attritionPerYear/totalInd
years <- seq(2004,2019)
# years
propAtrritionTable <- cbind(years,propAtrrition)
(propAtrritionTable)
```

```
##       years       propAtrrition
##  [1,]  2004 0.11596820809248555
##  [2,]  2005 0.18002351128947261
##  [3,]  2006 0.16151297625621203
##  [4,]  2007 0.20907050184529924
##  [5,]  2008 0.18760132787771172
##  [6,]  2009 0.16840324503056633
##  [7,]  2010 0.17346976744186046
##  [8,]  2011 0.15602553870710295
##  [9,]  2012 0.21096051856216852
## [10,]  2013 0.18614624218636430
## [11,]  2014 0.18920394569213408
## [12,]  2015 0.18961038961038962
## [13,]  2016 0.21249587261987746
## [14,]  2017 0.20849655801924466
## [15,]  2018 0.23640940917315711
## [16,]  2019 1.00000000000000000
```

```
  ######################
#testing 3.5 - Trying to look at other methods - not part of assignment
methodMergedMigration <- hhData %>% group_by(idmen,year) %>% summarise(
    migration = year == datent |
    if (year <= 2014){
    migration = year == myear # TRUE if year is equal to the move year
  }
  else if(year > 2014){
    migration = move == 2}
  ) # TRUE if they moved since last year (ie they moved this year)
```

```
## `summarise()` has grouped output by 'idmen'. You can override using the `.groups` argument.
```

```
#individuals in houses that have migrated (according to method2)




newAggData <- merge(aggData,methodMergedMigration, by = c("idmen","year"))
#add column of 0s to newAggData to track job movements
  Change <- rep(0,nrow(newAggData))
  newAggData <- cbind(newAggData,Change) #add column of zeros at the end which will then be used
for tracking job changes
migratedHouseData <- newAggData[ which(newAggData$migration == TRUE),]

#extract unique people
  uniqueIdind <- unique(migratedHouseData$idind)
  # length(uniqueIdind)

  #go through list
  for (i in 1:length(uniqueIdind)){
    ID <- uniqueIdind[i]
    #find indices corresponding to them
    index <- which(migratedHouseData[,"idind"] == ID)

    #check if empstat or profesison changed

    for (j in 1:length(index)){
      if (j > 1){

        #get index for years
        lastYearIndex <- index[j-1]
        thisYearIndex <- index[j]


        if(!is.na(migratedHouseData[lastYearIndex,"profession"]) &
           !is.na(migratedHouseData[thisYearIndex,"profession"])){

              #checked if not missing professions between this year and last year
          #check if not blank both years and if they are not the same (ie change occurred)

            if((migratedHouseData[lastYearIndex,"profession"] != "" &
              migratedHouseData[thisYearIndex,"profession"] != "") &
             (migratedHouseData[lastYearIndex,"profession"] !=
              migratedHouseData[thisYearIndex,"profession"])){

            migratedHouseData[thisYearIndex,"Change"] <- 1
          }
        }


          # check for change in empstat if profession doesnt change

        if (!is.na(migratedHouseData[lastYearIndex,"empstat"]) &
              !is.na(migratedHouseData[thisYearIndex,"empstat"])){
```

```
          if (migratedHouseData[lastYearIndex,"empstat"] != migratedHouseData[thisYearIndex,"emp
stat"]){
            migratedHouseData[thisYearIndex,"Change"] <- 1
          }
        }

      }

    }
  }
```