Factors that influence labor participation in 1976

Xiao Tan, section K3

3/28/2020

Introduction

This project is to explore the factors that influence labor participation in 1976, based on factors such as hours, age, kids, education, income, tax, college attendance in the family. We conducted this project to explore which factor would contribute most to the participaion of labor in 1976, especially for the wives and family issues. The variables I choose are more focused on wives and family factors. And here are some of my pre-analysis of the correlation between factors and labor participation, especially for the wives.

Education might be related to the labor force for several reasons. Firstly, people with more education will be more competitive in the labor market, so they are more likely to get a job. Also, people with more education have a higher payment, which will further attract them to join the labor market. The effect of age is mixed since it will influence the labor market because the older age might choose to retire, while the older age might have more working experience. Tax will affect the labor market by affecting the incomes. The higher the tax is, the less likely the worker will join the working market. The experience will positively affect labor force participation because the more experience they have, it means they are more competitive in the labor market. Also, the experience means they had once joined the labor market and are more likely to stay in the labor market. The young kids will negatively affect labor force participation because the young kids require a lot of care and attention, and the daycare and babysitting might be expensive for some family to afford, so the wives are more likely to stay at home to take care of young kids.

Introduction of the dataset:

This dataset is the Labor Force Participation Data, which is cross-section data originating from the 1976 Panel Study of Income Dynamics (PSID), based on data for the previous year, 1975. This data set is also known as the Mroz (1987) data.

Source of the data: Online complements to Greene (2003). Table F4.1.
http://pages.stern.nyu.edu/~wgreene/Text/tables/tablelist5.htm
(http://pages.stern.nyu.edu/~wgreene/Text/tables/tablelist5.htm)

References: Greene, W.H. (2003). Econometric Analysis, 5th edition. Upper Saddle River, NJ: Prentice Hall. McCullough, B.D. (2004). Some Details of Nonlinear Estimation. In: Altman, M., Gill, J., and McDonald, M.P.: Numerical Issues in Statistical Computing for the Social Scientist. Hoboken, NJ: John Wiley, Ch. 8, 199–218. Mroz, T.A. (1987). The Sensitivity of an Empirical Model of Married Women's Hours of Work to Economic and Statistical Assumptions. Econometrica, 55, 765–799. Winkelmann, R., and Boes, S. (2009). Analysis of Microdata, 2nd ed. Berlin and Heidelberg: Springer-Verlag. Wooldridge, J.M. (2002). Econometric Analysis of Cross-Section and Panel Data. Cambridge, MA: MIT Press.

Introduction of Variables:

Y outcome: **participation**: Did the individual participate in the labor force in 1975? (This is essentially wage > 0 or hours > 0.) The participation is a binary variable, with yes and no for answers.

X: factors:

In our project, we choose the education, age, tax, experience, youngkids for main discussion.

education: Wife's education in years.

age: Wife's age in years.

wage: Wife's average hourly wage, in 1975 dollars.

tax: Marginal tax rate facing the wife, and is taken from published federal tax tables (state and local

experience: Actual years of wife's previous labor market experience.

youngkids: Number of children less than 6 years old in household.

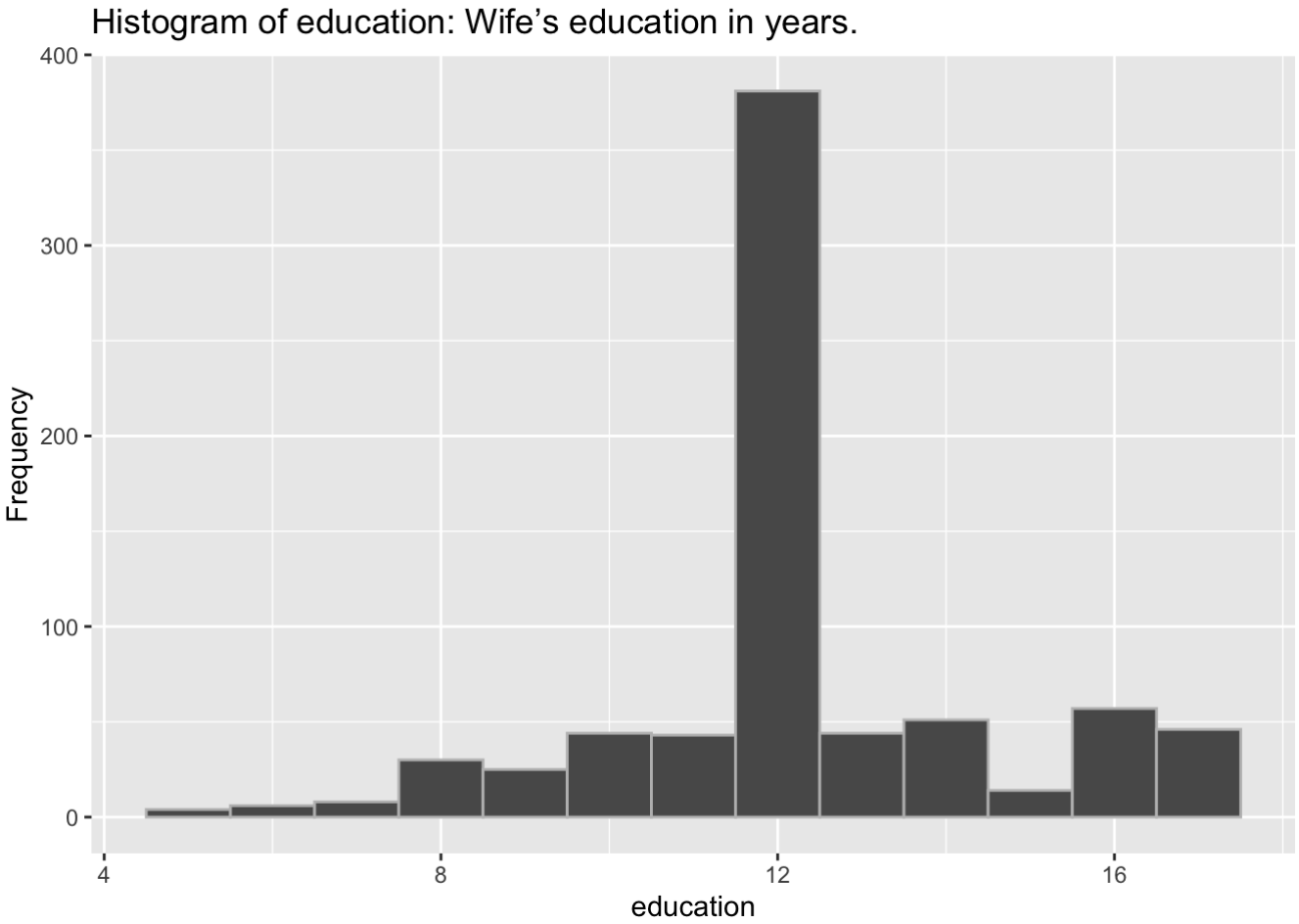
We will choose the *education, age, tax, experience*, and *youngkids* as our Xs to explain the Y. We want to analyze the labor force participant for wives, and we think the *education, age, tax experience and youngkids* would be the most influential factors intuitively.

Summary Statistics

Distribution of variables

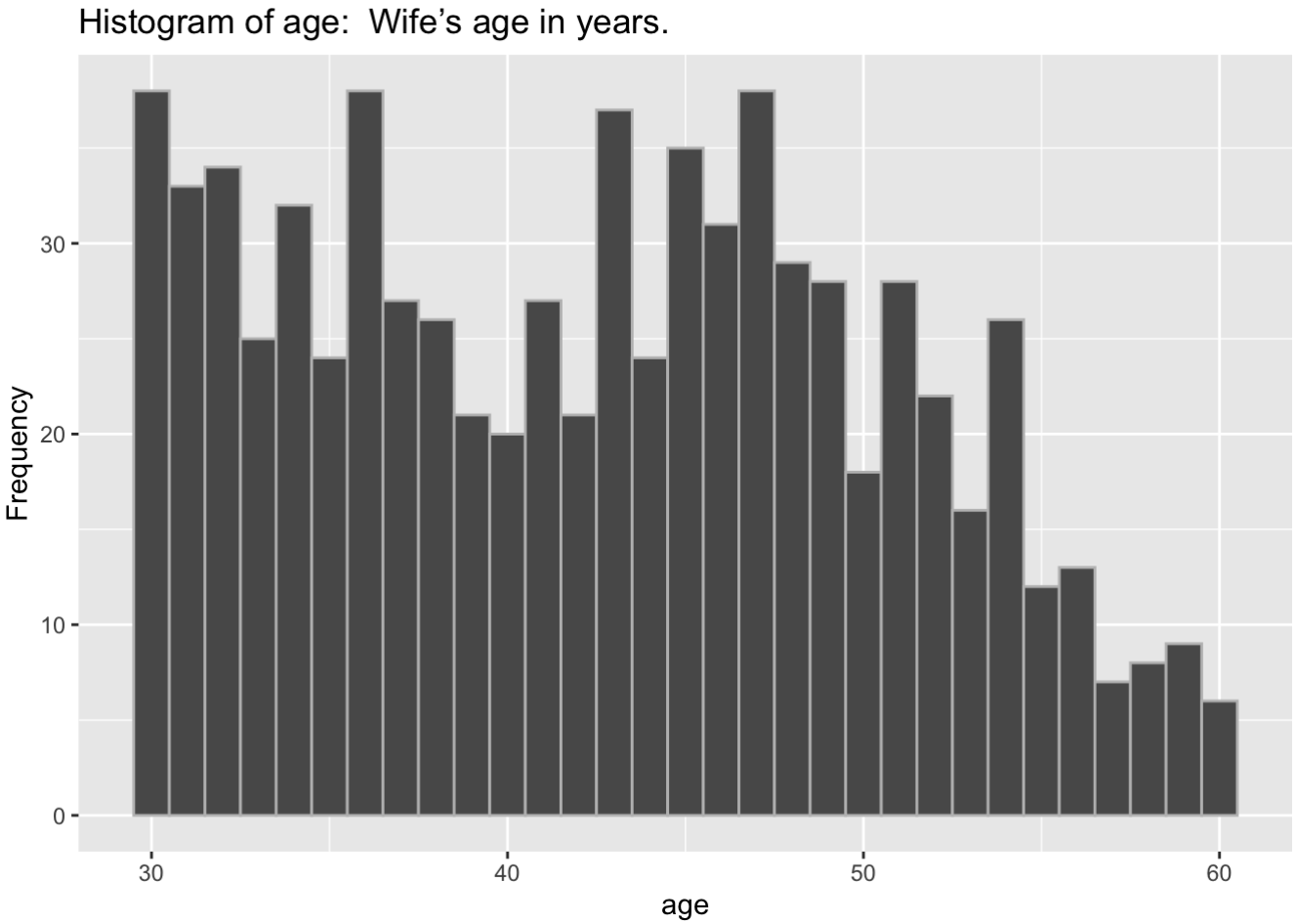
```
library(caret)
library(class)
library(glmnet)
library(tidyverse)
library(AER)
library(tree)
library(ISLR)
library(dplyr)
library(ggplot2)
library(MASS)
library(randomForest)
library(dplyr)
library(ggplot2)
library(AER)
library(glmnet)
library(tidyverse)
library(class)
library(gbm)
library(xgboost)
library(e1071)
library(ggplot2)
# find the datset of Labor Force Participation Data
library(AER)
data("PSID1976")
attach(PSID1976)
```

```
#education is continuous
ggplot(PSID1976) +
  labs(y="Frequency", x="education") +
  geom_histogram(aes(x=education),binwidth = 1, colour='grey') +
  ggtitle("Histogram of education: Wife's education in years.")
```



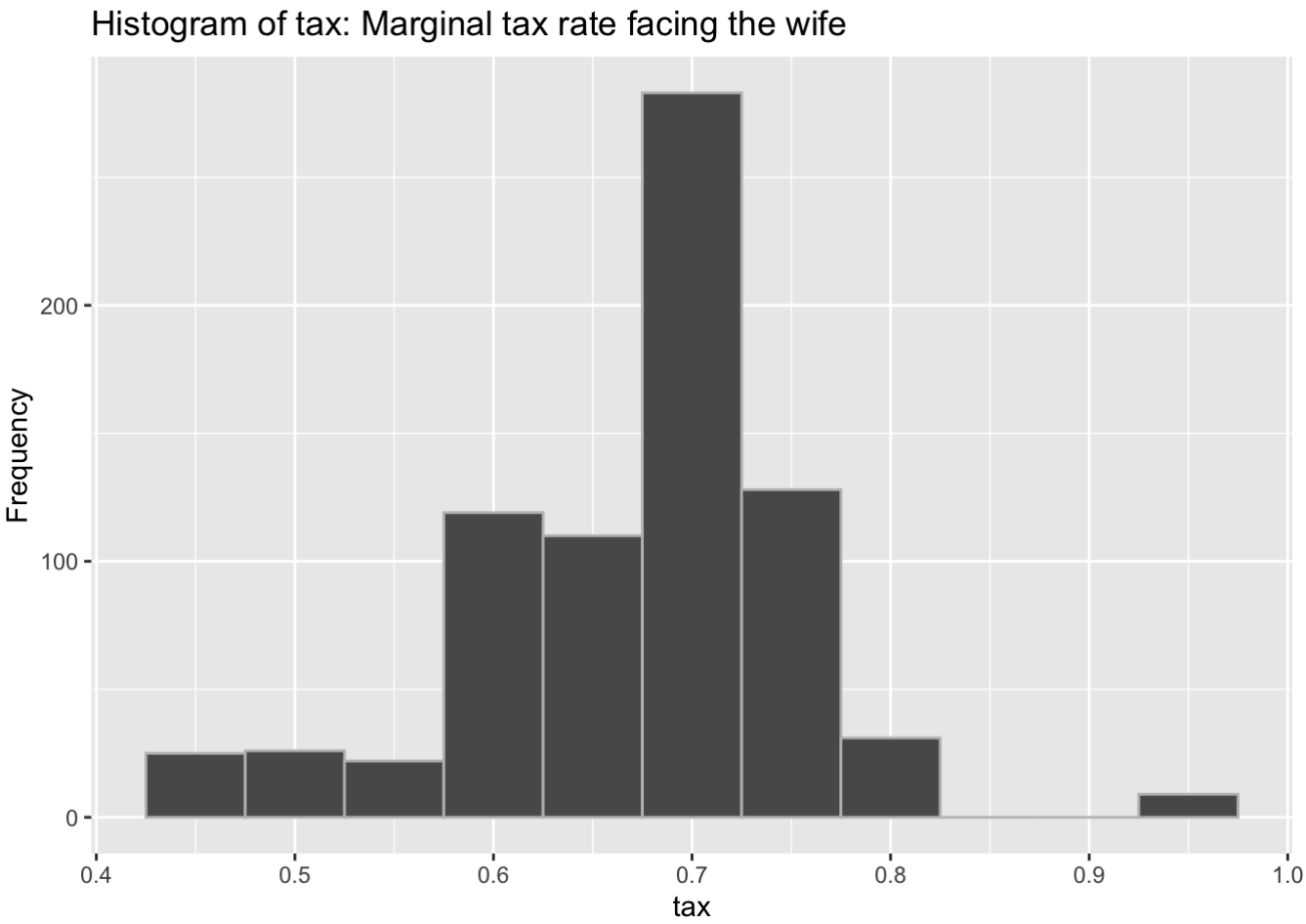
Education represents the wife’s education in years. The mode of education in the dataset is 12 years, which is probably high school graduation. And we have another increase in the frequency of education after that, 16 years, which is college graduation. This variable is similar to college attendance, so we only choose this one rather than bother of them. Also, we can see most of the wives have at least 8 years of education for most of the data points, and there is no data without any education.

```
#age is continuous
ggplot(PSID1976) +
  labs(y="Frequency", x="age") +
  geom_histogram(aes(x=age),binwidth = 1, colour='grey') +
  ggtitle("Histogram of age:  Wife’s age in years.")
```



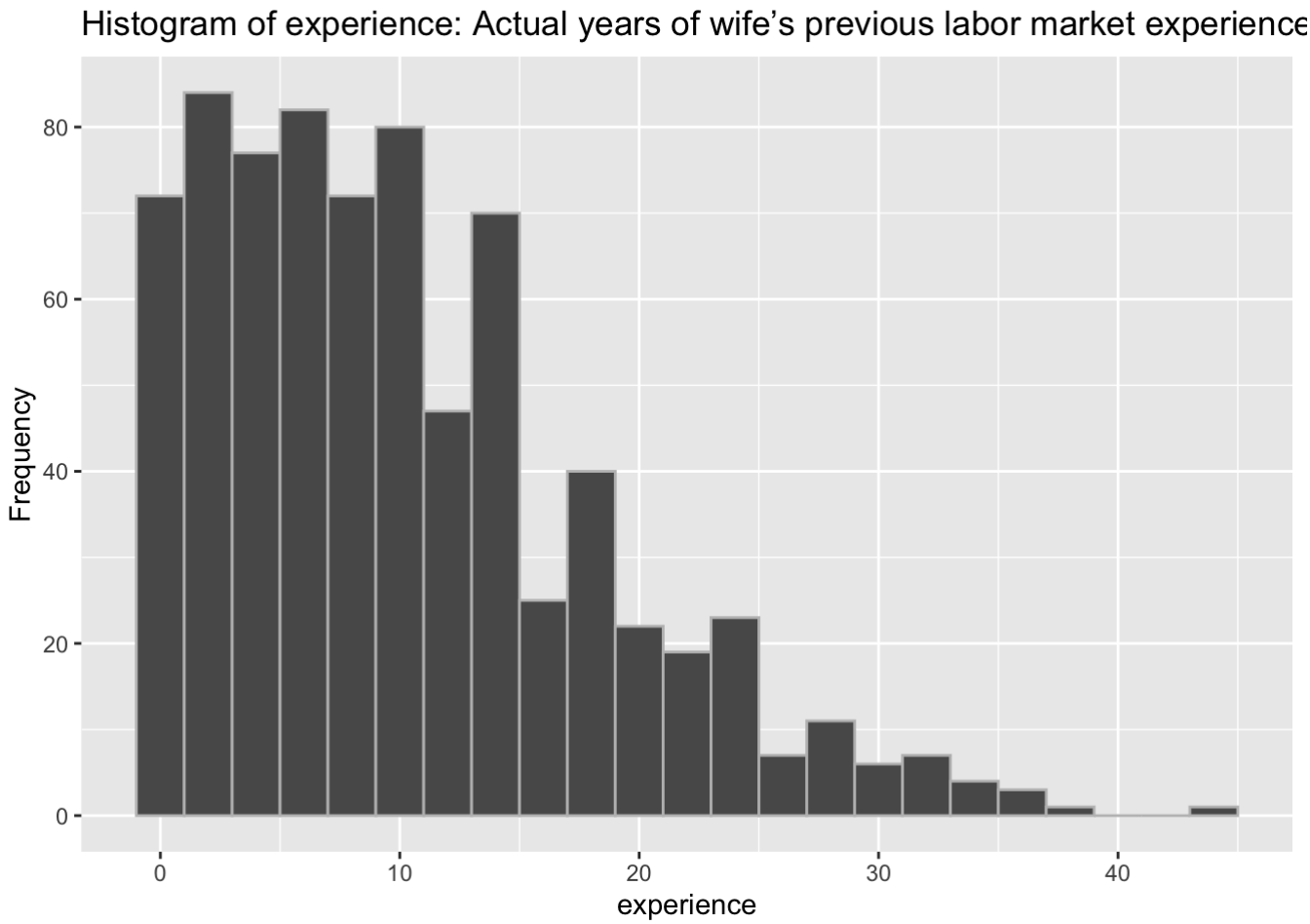
Age here represents the wife's age in years. We can see that most of the data points have at least 30 years and at most 60 years old, and the evenly distributed in this range of 30-50 and a decrease after that. This range is good this age range is after graduation from school and before retirement age. And this is also the normal age for married women.

```
#tax is continuous
ggplot(PSID1976) +
  labs(y="Frequency", x="tax") +
  geom_histogram(aes(x=tax),binwidth = 0.05, colour='grey') +
  ggtitle("Histogram of tax: Marginal tax rate facing the wife")
```



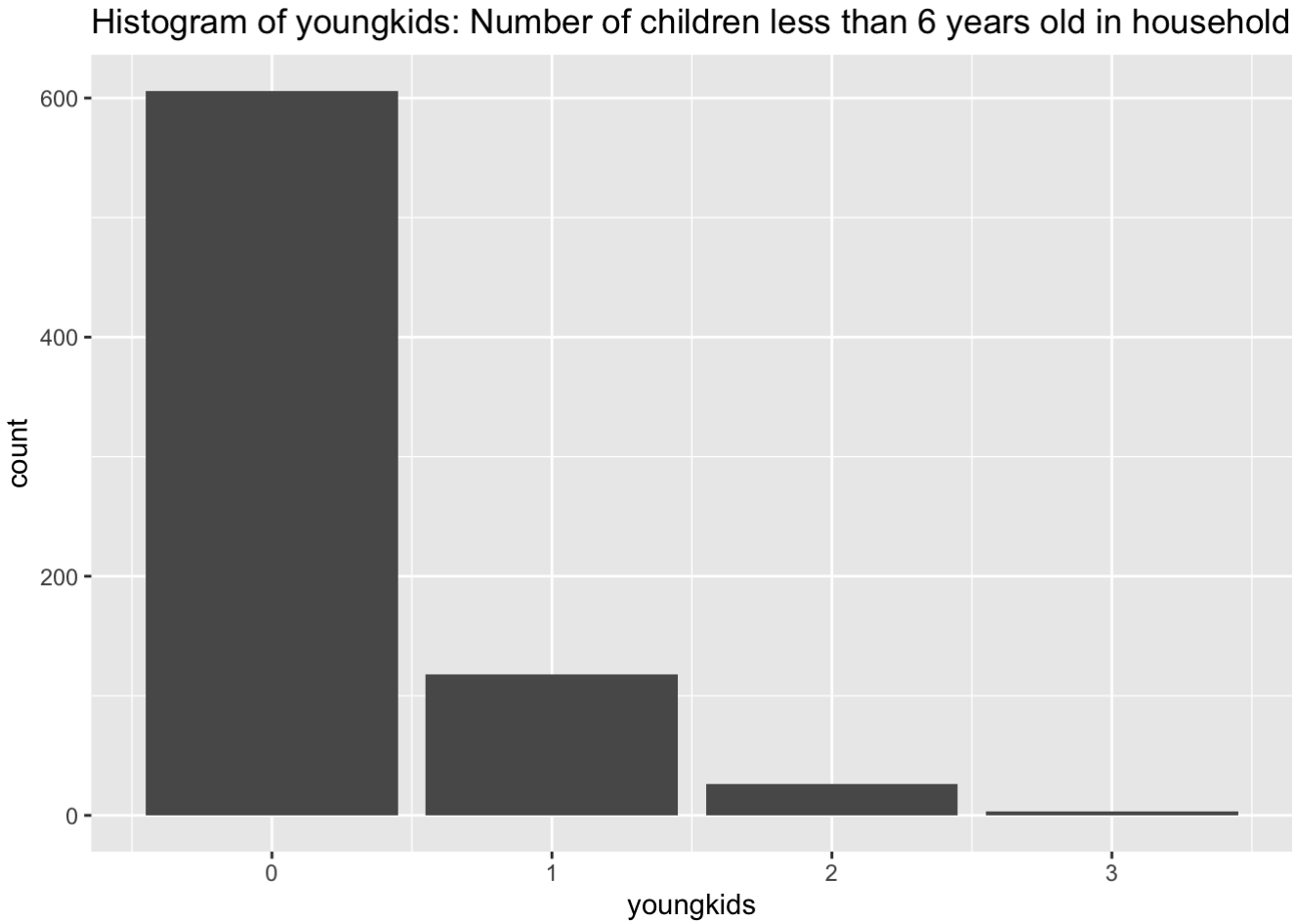
Tax here represents the marginal tax rate facing the wife and is taken from published federal tax tables (state and local). Most of the data points have a tax as more than 0.5 and the mode is around 0.7. This is probably the normal marginal tax rate of women in 1976.

```
#experience is continuous
ggplot(PSID1976) +
  labs(y="Frequency", x="experience") +
  geom_histogram(aes(x=experience),binwidth = 2, colour='grey') +
  ggtitle("Histogram of experience: Actual years of wife's previous labor market experie
nce")
```



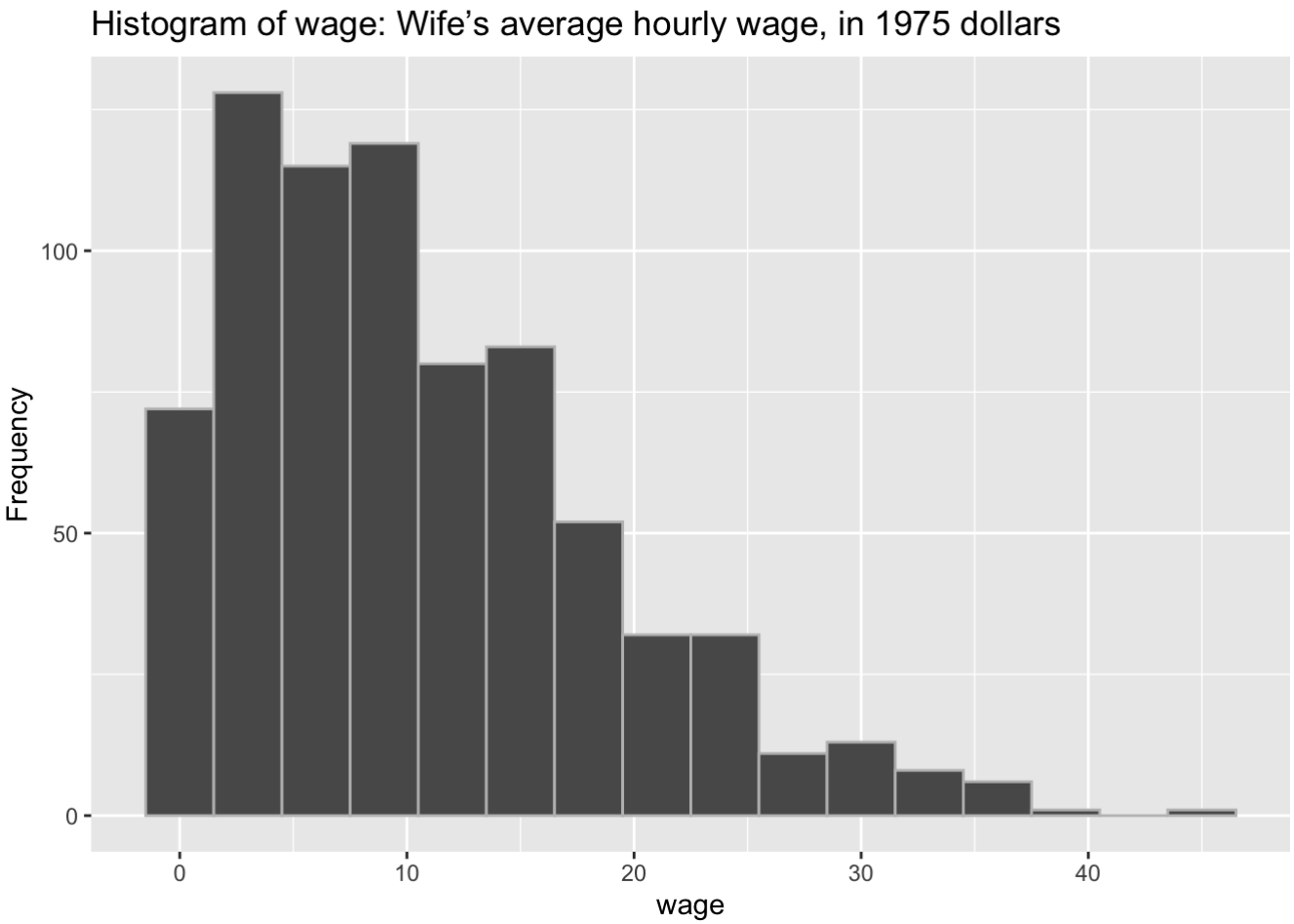
Experience here represents the actual years of the wife's previous labor market experience. We can see most of the wives have their experience range from 0 to 15 years. Also, there is a lot of them who have no working experience before. This is also caused by the normal age of working women.

```
#youngkids is discrete with the number 0,1,2,3
ggplot(PSID1976, aes(youngkids))+
  geom_bar() +
  labs(y="count")+
  ggtitle("Histogram of youngkids: Number of children less than 6 years old in household")
```



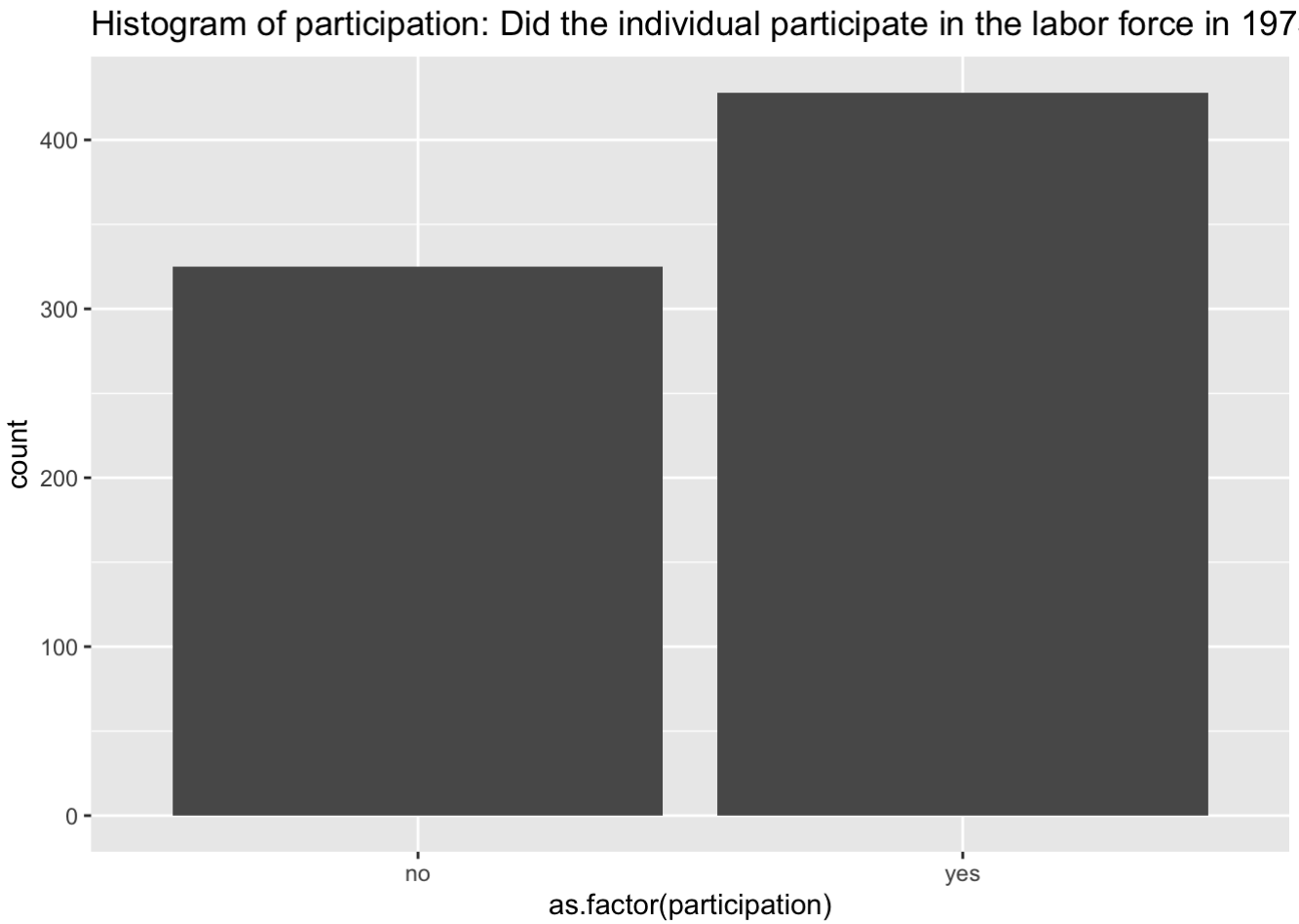
Youngkids here represent the number of children less than 6 years old in the household. Most of the families in the dataset do not have young kids. And about 150 data points have young kids at home. So the ratio of have young kids and do not have young kids is 4:1.

```
#wage is continuous
ggplot(PSID1976) +
  labs(y="Frequency", x="wage") +
  geom_histogram(aes(x=experience),binwidth = 3, colour='grey') +
  ggtitle("Histogram of wage: Wife's average hourly wage, in 1975 dollars")
```



wage here represent Wife’s average hourly wage, in 1975 dollars Most of the wifes have the wages around 5 to 15 at 1975, and very few of them have more than 30.

```
#Y participation is discrete
ggplot(PSID1976, aes(as.factor(participation)))+
  geom_bar() +
  labs(y="count")+
  ggtitle("Histogram of participation: Did the individual participate in the labor force
in 1975? ")
```

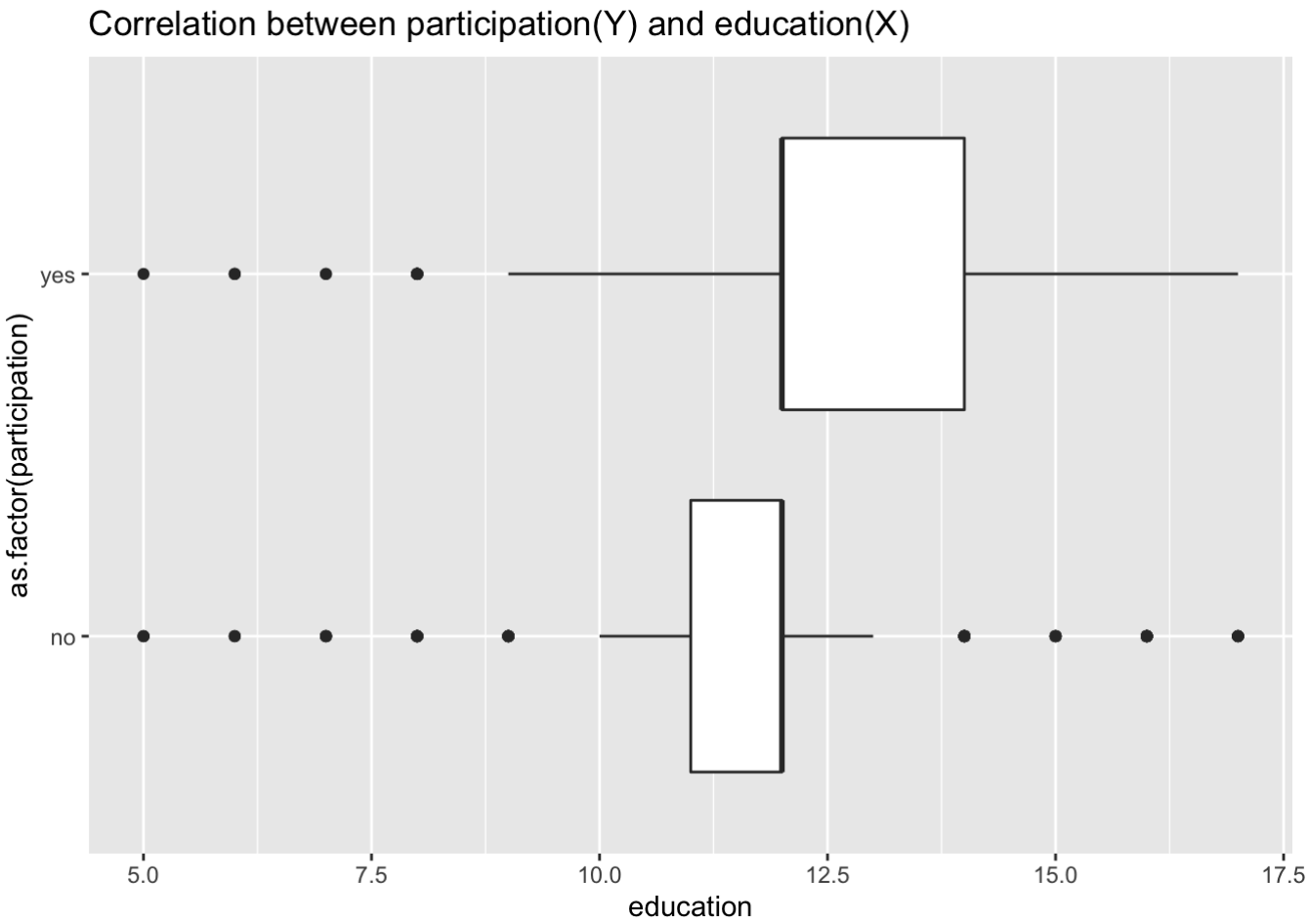


The participation results in our dataset are evenly distributed with a ratio of 7:9, which is close to 1:1. This is to say, the dataset is a fairly reasonable dataset with the output evenly distributed.

The correlation between Y and each selected X separately.

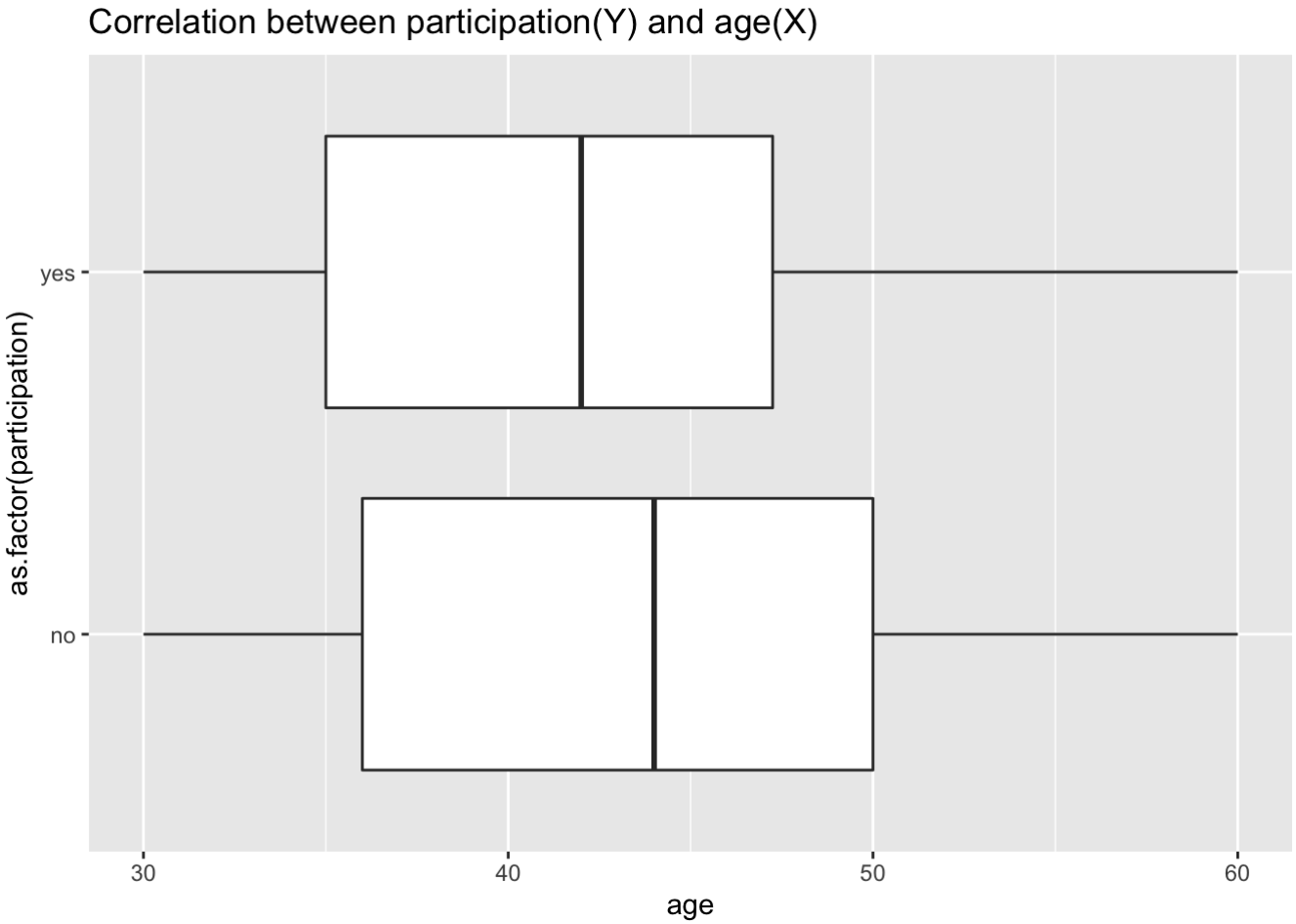
The *education, age, city, tax, experience, youngkids* as our Xs to explain the Y.

```
ggplot(PSID1976, aes(x=education, y=as.factor(participation))) +
  geom_boxplot() +
  ggtitle("Correlation between participation(Y) and education(X)")
```



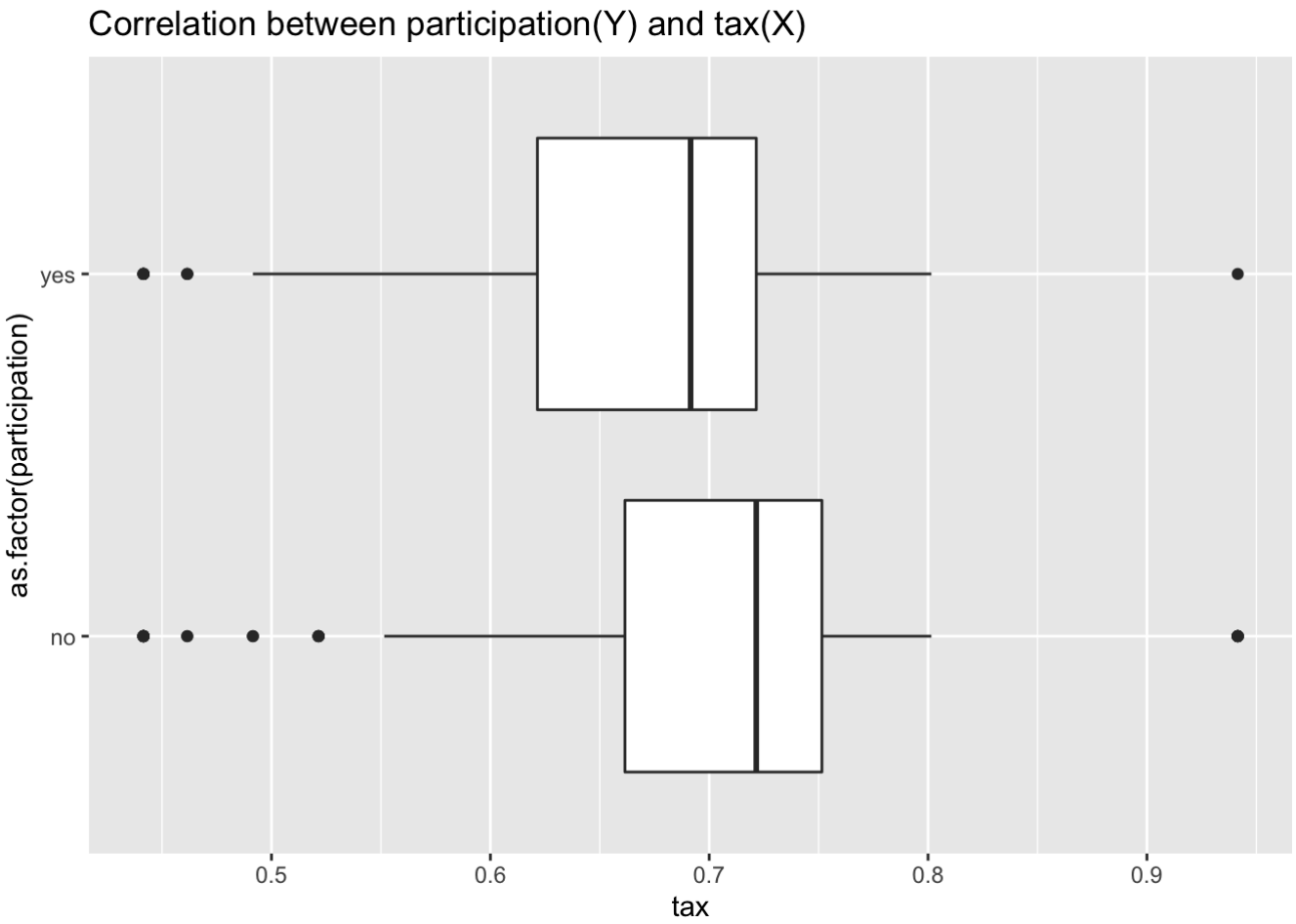
From the correlation between *participation* and *education*, we can see with more education, it is more likely to participate in the labor market in 1976. And differences between them are obvious because the 25-75 percentile of those two almost do not have a union. This means there is a strong correlation between *education* and *participation*.

```
ggplot(PSID1976, aes(x=age, y=as.factor(participation), color = age)) +
  geom_boxplot()+
  ggtitle("Correlation between participation(Y) and age(X)")+
  labs(
       color = "age")
```



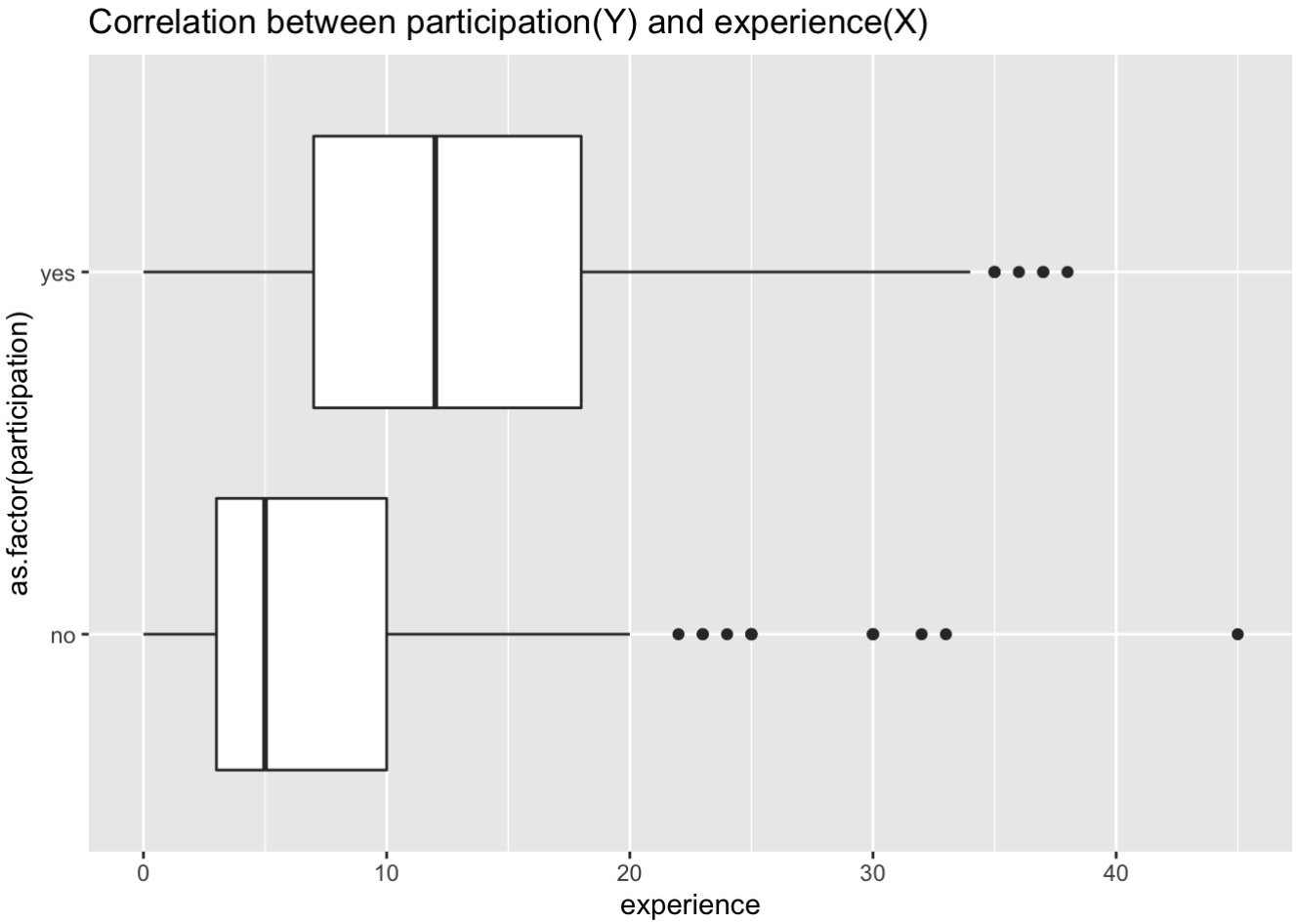
From the correlation between *participation* and *age*, we can see the younger age wives are more likely to participate in the labor market in 1976, although the differences between those two are not very obvious, which means the age is not a good indicator of labor force participation for wives. This means there is a weak correlation between *age* and *participation*.

```
ggplot(PSID1976, aes(x=tax, y=as.factor(participation))) +
  geom_boxplot() +
  ggtitle("Correlation between participation(Y) and tax(X)")
```



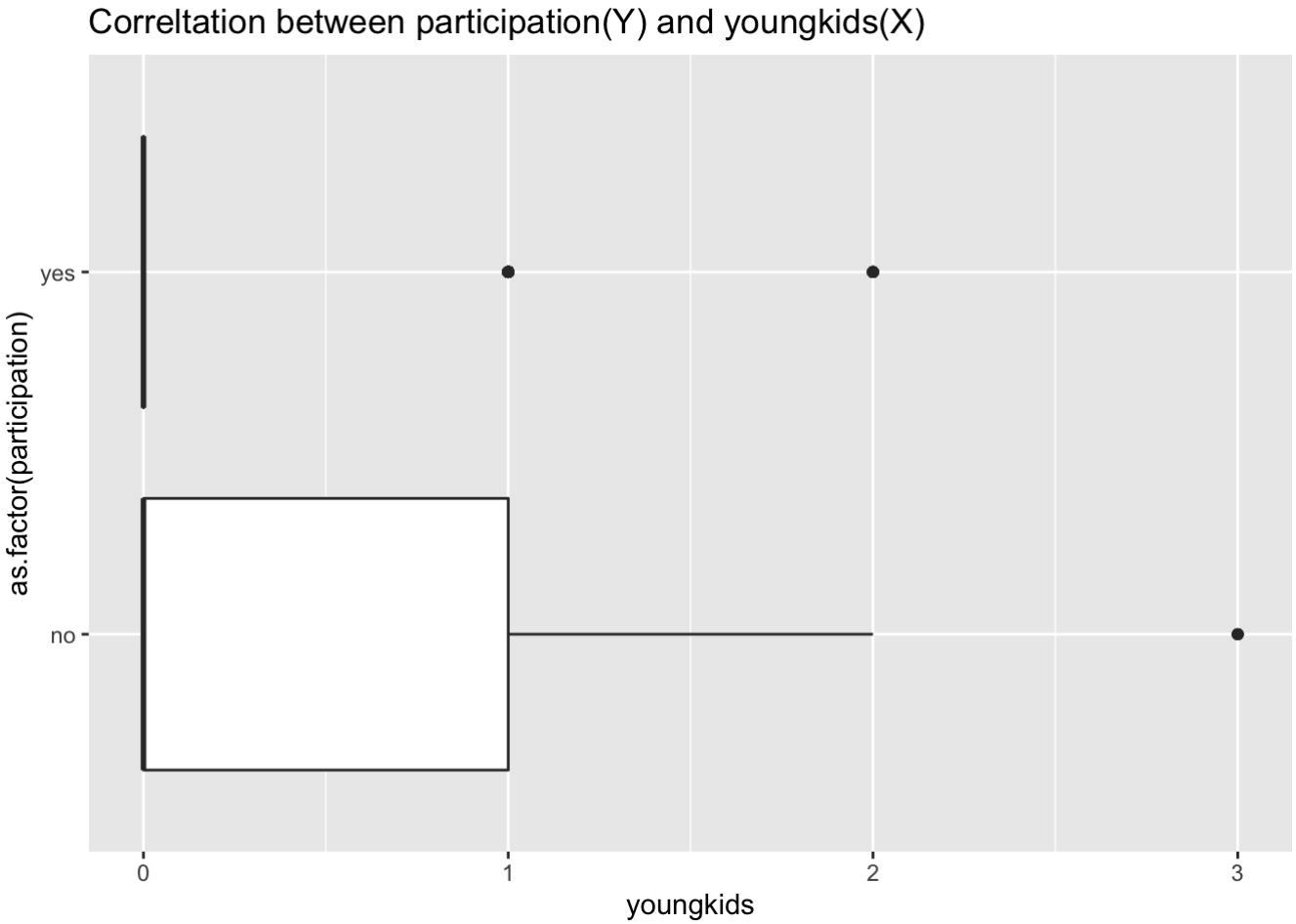
From the correlation of *participation* and *tax*, we can see with the lower the tax, it is more likely for the wives to participate in the labor market in 1976. This means there is a medium correlation between *tax* and *participation*.

```
ggplot(PSID1976, aes(x=experience, y=as.factor(participation))) +
  geom_boxplot() +
    ggtitle("Correlation between participation(Y) and experience(X)")
```



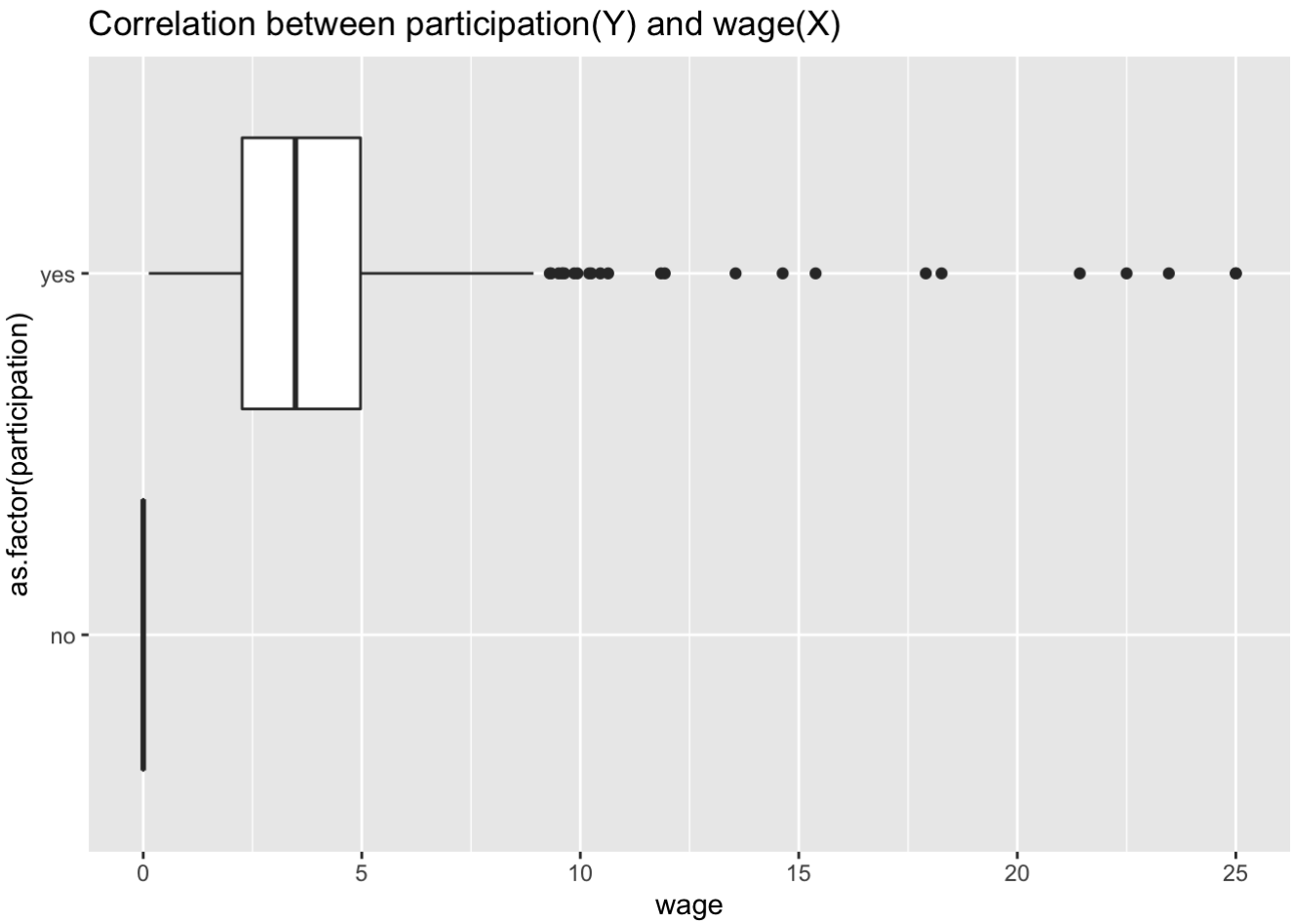
From the correlation of *participation* and *experience*, we can see with the more experience, it is more likely for the wives to participate in the labor market in 1976. And for those who enter the labor market, seldom with zero working experience, and most of them have more than five years of working experience. This means there is a strong correlation between *experience* and *participation*.

```
ggplot(PSID1976, aes(x= youngkids, y=as.factor(participation))) +
  geom_boxplot() +
  ggtitle("Correltation between participation(Y) and youngkids(X)")
```



From the correlation of *participation* and *youngkids*, we can see with the existence of young kids, most of the women will not attend the labor market. This means there is a strong correlation between *youngkids* and *participation*.

```
ggplot(PSID1976, aes(x=wage, y=as.factor(participation))) +
  geom_boxplot() +
    ggtitle("Correlation between participation(Y) and wage(X)")
```

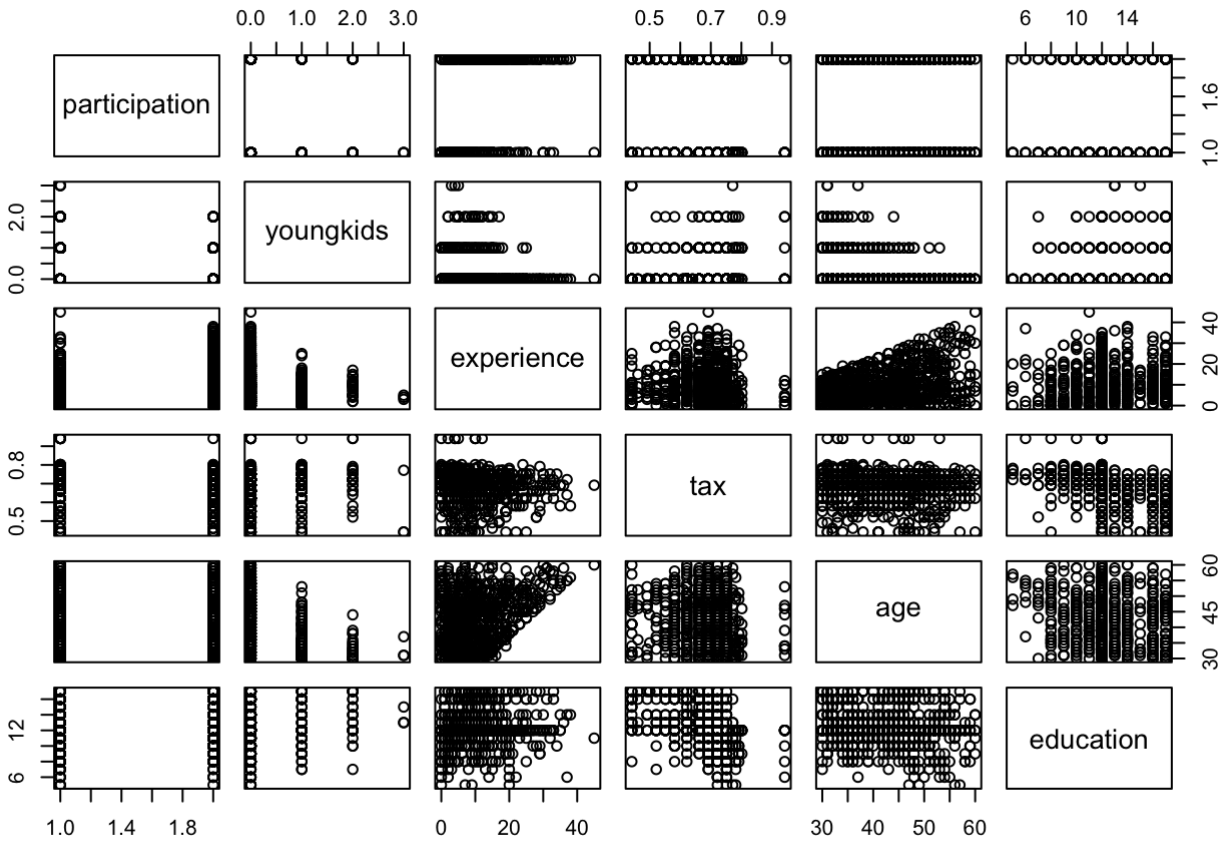


From the correlation of *participation* and *wage*, we can see with wifes will only attend the labor force market with wages.

Use the pairs to see the correlation between all the X and Ys

In this part, we use the pairs to see the correlation between all the X and Ys

```
pairs(~ participation + youngkids + experience + tax + age + education, PSID1976)
```



Because our Y(participation) is binary, it is hard to see the linear correlation in this plot, but we can see that age and experience have a strong correlation with each other. Besides that, other factors do not have strong correlations, so we are choosing the most independent variables in this data analysis. In conclusion, this is a good dataset to analyze the factors that influence labor participation in 1976.

Logistic Classification

Run different Logistic regressions with the same Y(participation) and compare their accuracy rates.

Firstly, we run the logistic regressions with all the variables above, which is our full model:

```
#the logistic regressions with Y(participation) and Xs(education, age, tax, experience,
 youngkids):
glm_5x = glm(as.factor(participation)~education+age+experience+tax+youngkids, data = PSI
D1976, family = binomial)
summary(glm_5x)
```

```
##
## Call:
## glm(formula = as.factor(participation) ~ education + age + experience +
##     tax + youngkids, family = binomial, data = PSID1976)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4681  -0.9161   0.4474   0.8797   2.0619
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  3.69691    1.27953   2.889 0.003861 **
## education    0.14744    0.04361   3.381 0.000723 ***
## age         -0.10269    0.01338  -7.674 1.67e-14 ***
## experience   0.12570    0.01328   9.467  < 2e-16 ***
## tax         -2.62710    1.14926  -2.286 0.022259 *
## youngkids   -1.41109    0.19810  -7.123 1.06e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1029.75  on 752  degrees of freedom
## Residual deviance:  813.62  on 747  degrees of freedom
## AIC: 825.62
##
## Number of Fisher Scoring iterations: 4
```

```
glm.probs= predict(glm_5x, PSID1976, type="response")
glm.pred=rep("no",753)
glm.pred[glm.probs>.5]="yes"
table(glm.pred,participation)
```

```
##         participation
## glm.pred  no yes
##      no  208  85
##      yes 117 343
```

```
mean(glm.pred==participation)
```

```
## [1] 0.7317397
```

In the model, we can see the 4 of 5 variables are very significant, the p-value from smallest to largest is:

experience < *age* < *youngkids* < *education* < *0.001* < *tax*.

We can see that the *tax* has the largest p-value, so we decide to delete tax:

```
#the logistic regressions with Y(participation) and X(education,age,experience,youngkid
s):
glm_4x = glm(as.factor(participation)~education+age+experience+youngkids, data = PSID197
6, family = binomial)
summary(glm_4x)
```

```
##
## Call:
## glm(formula = as.factor(participation) ~ education + age + experience +
##     youngkids, family = binomial, data = PSID1976)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4591  -0.9362   0.4530   0.8880   2.0283
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.31195    0.73482   1.785   0.0742 .
## education    0.18824    0.03997   4.710 2.48e-06 ***
## age         -0.09984    0.01328  -7.518 5.55e-14 ***
## experience   0.12403    0.01319   9.401  < 2e-16 ***
## youngkids   -1.44603    0.19836  -7.290 3.10e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1029.75  on 752  degrees of freedom
## Residual deviance:  818.94  on 748  degrees of freedom
## AIC: 828.94
##
## Number of Fisher Scoring iterations: 4
```

```
glm.probs= predict(glm_4x, PSID1976, type="response")
glm.pred=rep("no",753)
glm.pred[glm.probs>.5]="yes"
table(glm.pred,participation)
```

```
##         participation
## glm.pred  no yes
##      no  210  82
##      yes 115 346
```

```
mean(glm.pred==participation)
```

```
## [1] 0.7383798
```

Now we can see that, after deleting *tax*, all of the variables are significant, while the AIC value increases. The accurancy rate is 0.7383798.

The p-value from smallest to largest is:

experience < *age* < *youngkids* < *education* < *0.001*. (The same as model glm_5x).

We can also delete *education*, which has the largest p-value so far.

```
#the logistic regressions with Y(participation) and X(age,experience,youngkids):
glm_3x = glm(as.factor(participation)~age+experience+youngkids, data = PSID1976, family
 = binomial)
summary(glm_3x)
```

```
##
## Call:
## glm(formula = as.factor(participation) ~ age + experience + youngkids,
##     family = binomial, data = PSID1976)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5513  -0.9447   0.4717   0.9133   2.0378
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  3.66125    0.55201   6.633 3.30e-11 ***
## age         -0.10188    0.01302  -7.822 5.19e-15 ***
## experience   0.12593    0.01298   9.699  < 2e-16 ***
## youngkids   -1.32550    0.19170  -6.915 4.69e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1029.75  on 752  degrees of freedom
## Residual deviance:  842.65  on 749  degrees of freedom
## AIC: 850.65
##
## Number of Fisher Scoring iterations: 4
```

```
glm.probs= predict(glm_3x, PSID1976, type="response")
glm.pred=rep("no",753)
glm.pred[glm.probs>.5]="yes"
table(glm.pred,participation)
```

```
##         participation
## glm.pred  no yes
##      no  199  89
##      yes 126 339
```

```
mean(glm.pred==participation)
```

```
## [1] 0.7144754
```

After deleting *education*, the AIC still increases, and all the three variables have very small p-values. The accurancy rate is 0.7144754.

The p-value from smallest to largest is: *experience* < *age* < *youngkids*

We can delete *youngkids*, the largest p-value so far.

```
#the logistic regressions with Y(participation) and X(age,experience):
glm_2x = glm(as.factor(participation)~age+experience, data = PSID1976, family = binomia
l)
summary(glm_2x)
```

```
##
## Call:
## glm(formula = as.factor(participation) ~ age + experience, family = binomial,
##     data = PSID1976)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6850  -1.0390   0.5714   0.9768   1.9476
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.56783    0.43840   3.576 0.000349 ***
## age         -0.06003    0.01087  -5.520 3.38e-08 ***
## experience   0.12469    0.01258   9.909  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1029.7  on 752  degrees of freedom
## Residual deviance:  898.7  on 750  degrees of freedom
## AIC: 904.7
##
## Number of Fisher Scoring iterations: 4
```

```
glm.probs= predict(glm_2x, PSID1976, type="response")
glm.pred=rep("no",753)
glm.pred[glm.probs>.5]="yes"
table(glm.pred,participation)
```

```
##         participation
## glm.pred  no yes
##      no  186 113
##      yes 139 315
```

```
mean(glm.pred==participation)
```

```
## [1] 0.6653386
```

We can see that the AIC is still increasing. The accurancy rate is 0.6653386.

So, now I would like to run the Logistic regressions for each of the variables separately, in the order of the p-value of in model glm_5x from smallest to largest (*experience* < *age* < *youngkids* < *education* < *0.001* < *tax*) :

```
#the logistic regressions with Y(participation) and experience:
glm_experience = glm(as.factor(participation)~experience, data = PSID1976, family = bino
mial)
summary(glm_experience)
```

```
##
## Call:
## glm(formula = as.factor(participation) ~ experience, family = binomial,
##     data = PSID1976)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8234  -1.0761   0.5643   1.0175   1.5166
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.76921    0.13460  -5.715  1.1e-08 ***
## experience   0.10525    0.01193   8.823  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1029.7  on 752  degrees of freedom
## Residual deviance:  931.2  on 751  degrees of freedom
## AIC: 935.2
##
## Number of Fisher Scoring iterations: 4
```

```
glm.probs= predict(glm_experience, PSID1976, type="response")
glm.pred=rep("no",753)
glm.pred[glm.probs>.5]="yes"
table(glm.pred,participation)
```

```
##         participation
## glm.pred  no yes
##      no  196 119
##      yes 129 309
```

```
mean(glm.pred==participation)
```

```
## [1] 0.6706507
```

The accurancy rate is 0.6706507.

```
#the logistic regressions with Y(participation) and age:
glm_age = glm(as.factor(participation)~age, data = PSID1976, family = binomial)
summary(glm_age)
```

```
##
## Call:
## glm(formula = as.factor(participation) ~ age, family = binomial,
##     data = PSID1976)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.4093  -1.2665   0.9776   1.0660   1.2097
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.136360   0.398338   2.853  0.00433 **
## age         -0.020201   0.009165  -2.204  0.02752 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1029.7  on 752  degrees of freedom
## Residual deviance: 1024.9  on 751  degrees of freedom
## AIC: 1028.9
##
## Number of Fisher Scoring iterations: 4
```

```
glm.probs= predict(glm_age, PSID1976, type="response")
glm.pred=rep("no",753)
glm.pred[glm.probs>.5]="yes"
table(glm.pred,participation)
```

```
##         participation
## glm.pred  no yes
##      no   16  14
##      yes 309 414
```

```
mean(glm.pred==participation)
```

```
## [1] 0.5710491
```

The accurancy rate is 0.5710491.

```
#the logistic regressions with Y(participation) and youngkids:
glm_youngkids = glm(as.factor(participation)~youngkids, data = PSID1976, family = binomi
al)
summary(glm_youngkids)
```

```
##
## Call:
## glm(formula = as.factor(participation) ~ youngkids, family = binomial,
##     data = PSID1976)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.3869  -1.3869   0.9815   0.9815   1.7392
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.48006    0.08265   5.808 6.32e-09 ***
## youngkids   -0.87179    0.15705  -5.551 2.84e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1029.75  on 752  degrees of freedom
## Residual deviance:  994.75  on 751  degrees of freedom
## AIC: 998.75
##
## Number of Fisher Scoring iterations: 4
```

```
glm.probs= predict(glm_youngkids, PSID1976, type="response")
glm.pred=rep("no",753)
glm.pred[glm.probs>.5]="yes"
table(glm.pred,participation)
```

```
##         participation
## glm.pred  no yes
##      no   94  53
##      yes 231 375
```

```
mean(glm.pred==participation)
```

```
## [1] 0.622842
```

The accurancy rate is 0.622842.

```
#the logistic regressions with Y(participation) and education:
glm_education = glm(as.factor(participation)~education, data = PSID1976, family = binomi
al)
summary(glm_education)
```

```
##
## Call:
## glm(formula = as.factor(participation) ~ education, family = binomial,
##     data = PSID1976)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.668  -1.279   0.815   1.079   1.613
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.85199    0.42828  -4.324 1.53e-05 ***
## education    0.17398    0.03465   5.022 5.12e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1029.7  on 752  degrees of freedom
## Residual deviance: 1002.7  on 751  degrees of freedom
## AIC: 1006.7
##
## Number of Fisher Scoring iterations: 4
```

```
glm.probs= predict(glm_education, PSID1976, type="response")
glm.pred=rep("no",753)
glm.pred[glm.probs>.5]="yes"
table(glm.pred,participation)
```

```
##         participation
## glm.pred  no yes
##      no   67  50
##      yes 258 378
```

```
mean(glm.pred==participation)
```

```
## [1] 0.5909695
```

The accurancy rate is 0.5909695.

```
#the logistic regressions with Y(participation) and tax:
glm_tax = glm(as.factor(participation)~tax, data = PSID1976, family = binomial)
summary(glm_tax)
```

```
##
## Call:
## glm(formula = as.factor(participation) ~ tax, family = binomial,
##     data = PSID1976)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6896  -1.2316   0.9204   1.0788   1.4764
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   2.7715     0.6446   4.300 1.71e-05 ***
## tax          -3.6659     0.9371  -3.912 9.16e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1029.7  on 752  degrees of freedom
## Residual deviance: 1013.6  on 751  degrees of freedom
## AIC: 1017.6
##
## Number of Fisher Scoring iterations: 4
```

```
glm.probs= predict(glm_tax, PSID1976, type="response")
glm.pred=rep("no",753)
glm.pred[glm.probs>.5]="yes"
table(glm.pred,participation)
```

```
##         participation
## glm.pred  no yes
##      no   38  17
##      yes 287 411
```

```
mean(glm.pred==participation)
```

```
## [1] 0.5962815
```

The accurancy rate is 0.5962815.

By comparing the AIC of each variable and Y, we find that the AIC of *age* is highest, which is beyond our expectation. So we run another model by deleting *age* from our full model(glm_5x) to check if it is better if we delete the variable *age*:

```
#the logistic regressions with Y(participation) and Xs(education,  tax, experience, youn
gkids):
glm_no_age = glm(as.factor(participation)~education+tax+experience+youngkids, data = PSI
D1976, family = binomial)
summary(glm_no_age)
```

```
##
## Call:
## glm(formula = as.factor(participation) ~ education + tax + experience +
##     youngkids, family = binomial, data = PSID1976)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6914  -1.0351   0.5362   0.9569   2.0532
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.50102    1.04784  -1.432    0.152
## education    0.17548    0.04143   4.236 2.27e-05 ***
## tax         -1.68537    1.09275  -1.542    0.123
## experience   0.09624    0.01224   7.864 3.72e-15 ***
## youngkids   -0.75643    0.16564  -4.567 4.95e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1029.75  on 752  degrees of freedom
## Residual deviance:  881.03  on 748  degrees of freedom
## AIC: 891.03
##
## Number of Fisher Scoring iterations: 4
```

```
glm.probs= predict(glm_no_age, PSID1976, type="response")
glm.pred=rep("no",753)
glm.pred[glm.probs>.5]="yes"
table(glm.pred,participation)
```

```
##         participation
## glm.pred  no yes
##      no  188  96
##      yes 137 332
```

```
mean(glm.pred==participation)
```

```
## [1] 0.690571
```

The accurancy rate is 0.690571.

The AIC of this model is not higher than our full model, so we can keep the variable *age*.

In conclusion, we ran 10 logistic regressions.

For the first 9 models, their Xs and AIC are as follows:

Xs variables in the models	**AIC value**	**accurancy rate**

Xs variables in the models	AIC value	accurancy rate
education+age+tax+experience+youngkids	AIC: 825.62;	0.7317397;
education+age+experience+youngkids	AIC: 828.94;	0.7383798;
age+experience+youngkids	AIC: 850.65;	0.7144754;
age+experience	AIC: 904.7;	0.6653386;
experience	AIC: 935.2;	0.6706507;
age	AIC: 1028.9;	0.5710491;
youngkids	AIC: 998.75;	0.622842;
education	AIC: 1006.7;	0.5909695;
tax	AIC: 1017.6;	0.5962815;

From the table above, We can see that the AIC is increasing as we delete the variables from our full model, which means the full model has the highest accuracy. Also, from the accurancy rate, we can see that the full modeln and the model without tax have the higest accurancy rate. Compared with AIC, we will choose the full model to analyze.

We also compare the accuracy rates and AIC of each variable separately. While as we compare the accuracy rates of each variable, we find that the AIC of *age* is even unusually high. In the full model, the p-value of *age* is the second smallest, while in separate models, the AIC of *age* is the largest. To solve this, we run another model(glm_no_age) by deleting *age* from the full model. We can see that the AIC of model glm_no_age has an AIC of 891.03, which is still a rather high number. Also, in terms of the accurancy rate, the full model is good. So, we can keep *age* in our final model.

In general, the relationships make sense, the more variables can generate a higher rate of accuracy.

Analysis and interpretation of our best regression model

In this section, we will analyze our best model: Y = *participation* X = *education, age, tax, experience, youngkids*

```
summary(glm_5x)
```

```
##
## Call:
## glm(formula = as.factor(participation) ~ education + age + experience +
##     tax + youngkids, family = binomial, data = PSID1976)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4681  -0.9161   0.4474   0.8797   2.0619
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  3.69691    1.27953   2.889 0.003861 **
## education    0.14744    0.04361   3.381 0.000723 ***
## age         -0.10269    0.01338  -7.674 1.67e-14 ***
## experience   0.12570    0.01328   9.467  < 2e-16 ***
## tax         -2.62710    1.14926  -2.286 0.022259 *
## youngkids   -1.41109    0.19810  -7.123 1.06e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1029.75  on 752  degrees of freedom
## Residual deviance:  813.62  on 747  degrees of freedom
## AIC: 825.62
##
## Number of Fisher Scoring iterations: 4
```

From the intercepts of the model, we can see that the absolute value of coefficients of the *tax* and *youngkids* are the largest, while the p value of *tax* is very large, so *youngkids*will have the largest effect on Y. And the p-values of experience and age are the smallest, which means they are very significant.

We can interpret the coefficients in this way:

Holding other variables constant, if the *education* increases in one unit, which means the wife has one more year of education, the odds ratio of labor participation will increase at about 0.14 percent.

Holding other variables constant, if the *age* increases in one unit (one year), which means the wife is one year older, the odds ratio of labor participation will decrease at about 0.10 percent.

Holding other variables constant, if the *tax* increases in one unit, which is the marginal tax rate facing the wife, the odds ratio of labor participation will decrease at about 2.6 percent.

Holding other variables constant, if the *experience* increases in one unit, which means the actual years of wife's previous labor market experience increase one year, the odds ratio of labor participation will increase at about 0.1257 percent..

Holding other variables constant, if the *youngkids* increases in one unit, which means one more child less than 6 years old in the household, the odds ratio of labor participation will decrease at about 1.41 percent.

According to the p-value of each variable, *education, age, experience, youngkids* have p-values lower than 0.001, so they are significantly different from zero, with confidence level larger than 99%.

** The true positive and false positive rates of this logistic regressions model**

```
glm.probs= predict(glm_5x, PSID1976, type="response")
glm.pred=rep("no",753)
glm.pred[glm.probs>.5]="yes"
table(glm.pred,participation)
```

```
##         participation
## glm.pred  no yes
##      no  208  85
##      yes 117 343
```

```
mean(glm.pred==participation)
```

```
## [1] 0.7317397
```

```
true_positive = 343/(343+85)
true_positive
```

```
## [1] 0.8014019
```

```
false_positive = 117/(117+208)
false_positive
```

```
## [1] 0.36
```

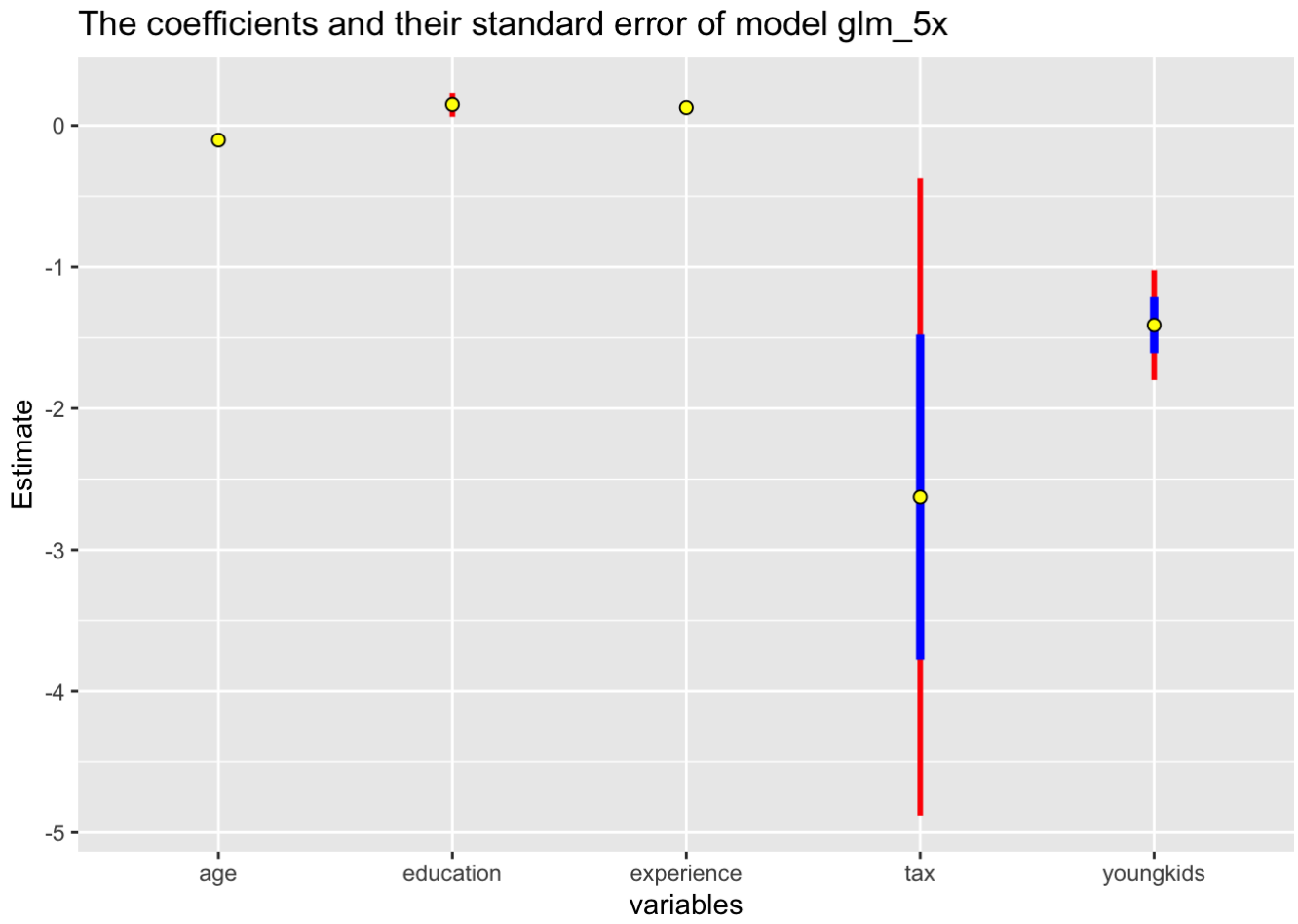
So, true_positive = 0.8014019, which means the model correctly predicts the positive class for 80.14019%. And false_positive = 0.36, which means the model incorrectly predicts the positive class for 36%.

This regression has a fairly good to predict labor participation in 1976 on the factors *education, age, tax, experience*, and *youngkids*.

Plot the estimated coefficients and their standard error.

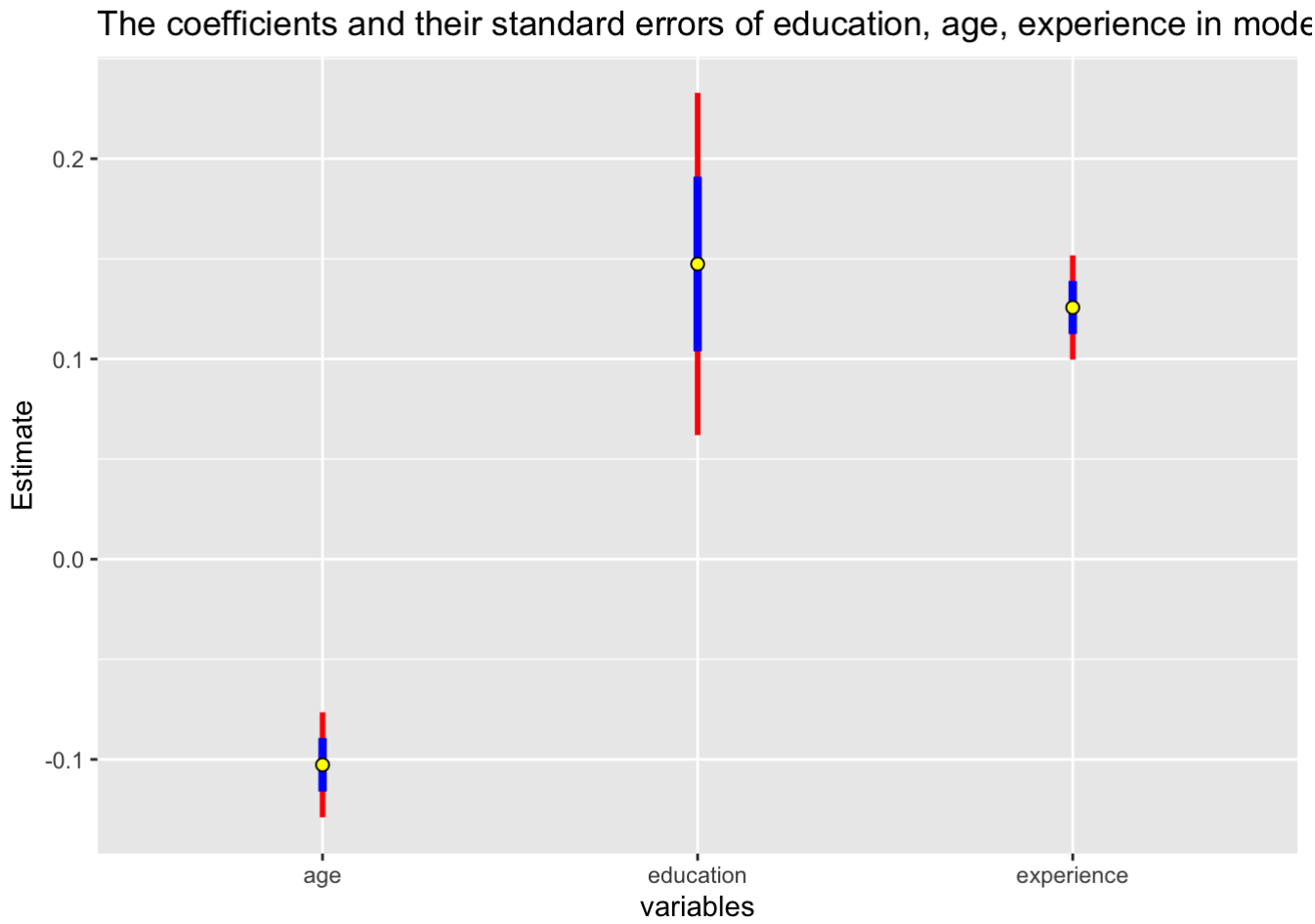
Now we plot the coefficients and their standard error in the model glm_5x:

```
coefs = as.data.frame(summary(glm_5x)$coefficients[-1,1:2]) # -1 is to exclude the inter
cept
names(coefs)[2] = "se"
coefs$vars = rownames(coefs)
ggplot(coefs, aes(vars, Estimate)) +
geom_errorbar(aes(ymin=Estimate - 1.96*se, ymax=Estimate + 1.96*se), lwd=1, colour="red"
, width=0) +
geom_errorbar(aes(ymin=Estimate - se, ymax=Estimate + se), lwd=1.5, colour="blue", width
=0) +
geom_point(size=2, pch=21, fill="yellow")+
labs(title = "The coefficients and their standard error of model glm_5x ", x ="variable
s", y ="Estimate")
```



Since it is hard to see the first three variables, we plot variables from 1 to 3, *education, age, experience*, again.

```
coefs = as.data.frame(summary(glm_5x)$coefficients[2:4,1:2]) # -1 is to exclude the inte
rcept
names(coefs)[2] = "se"
coefs$vars = rownames(coefs)
ggplot(coefs, aes(vars, Estimate)) +
geom_errorbar(aes(ymin=Estimate - 1.96*se, ymax=Estimate + 1.96*se), lwd=1, colour="red"
, width=0) +
geom_errorbar(aes(ymin=Estimate - se, ymax=Estimate + se), lwd=1.5, colour="blue", width
=0) +
geom_point(size=2, pch=21, fill="yellow")+
labs(title = "The coefficients and their standard errors of education, age, experience i
n model glm_5x ", x ="variables", y ="Estimate")
```



We can see that the confidence interval of those variables does not contain 0. So, all the coefficients are significant.

Probit Classification

```
glm.fits=glm(as.factor(participation)~education+age+experience+tax+youngkids,data=PSID19
76,family = binomial(link = "probit"))
glm.probs=predict(glm.fits,PSID1976,type="response")
glm.pred[glm.probs>.5]="yes"
glm.pred[glm.probs<=.5]="no"
table(glm.pred,PSID1976$participation)
```

```
##
## glm.pred  no yes
##      no  209  84
##      yes 116 344
```

```
mean(glm.pred==PSID1976$participation)
```

```
## [1] 0.7343958
```

```
true_positive = 344/(344+116)
true_positive
```

```
## [1] 0.7478261
```

```
false_positive = 84/(84+209)
false_positive
```

```
## [1] 0.2866894
```

The accuracy is 0.7343958 and the accuracy of the Logistic regression is 0.7317397, so it is slightly higher than Logistic regression.

And the true_positive is 0.7478261 and the true_positive of Logistic regression is 0.7456522. So it is also slightly higher than Logistic regression in terms of rate of true_positive.

In general, the accuracy of Probit regression and Logistic regression are very close. Probit regression's accuracy rate is slightly higher than the Logistic regression's accuracy rate.

Ridge:

If alpha = 0 then a ridge regression model is fit, and if alpha = 1 then a lasso model is fit. Firstly, we will discuss the Ridge model:

Tune the model: Use cross-validation to find the flexibly of the model (lamda)：

```
options(warn =-1)

PSID1976$participation <- ifelse(PSID1976$participation=='yes',1,0)
library(class)
library(glmnet)
library(tidyverse)
set.seed(233)
index <- sample(1:nrow(PSID1976),round(0.7*nrow(PSID1976)))
train <- as.data.frame(PSID1976[index,])
test <- as.data.frame(PSID1976[-index,])


x = model.matrix(participation~., PSID1976)[,-1]
y = PSID1976$participation

x_train = model.matrix(participation~., train)[,-1]
y_train = train$participation

x_test = model.matrix(participation~., test)[,-1]
y_test = test$participation
grid = 10^seq(10, -2, length = 100)
```

```
ridge_mod = glmnet(x_train, y_train, alpha=0, lambda = grid)
set.seed(1)
cv.out_ridge = cv.glmnet(x_train, y_train, alpha = 0) # Fit ridge regression model on tr
aining data
cv.out_ridge
```

```
##
## Call:  cv.glmnet(x = x_train, y = y_train, alpha = 0)
##
## Measure: Mean-Squared Error
##
##      Lambda Measure       SE Nonzero
## min 0.03708 0.08035 0.004709      20
## 1se 0.19789 0.08477 0.004601      20
```

We use cross-validation to tune the model. The min value, 0.03708, is the lambda, which is the one which minimizes out-of-sample loss in CV. The 1se value, 0.19789, is the lambda, which is the largest lambda within 1 standard error of the minimum lambda.

The lamda which is within 1 standard error of the minimum lamda:

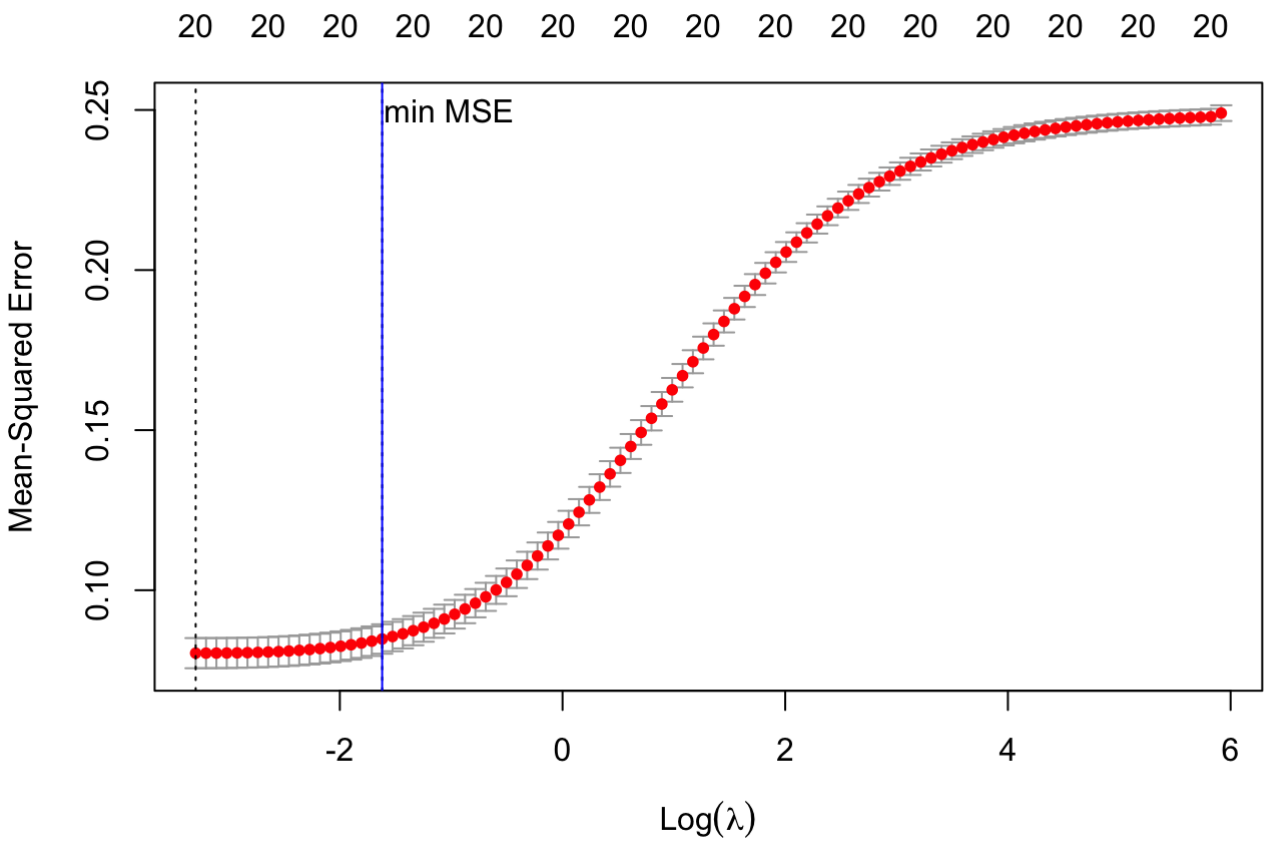
```
seleclam_ridge = cv.out_ridge$lambda.1se  # Select lamda that minimizes training MSE, wh
ich also statisfy the within 1se of the min lambda
seleclam_ridge
```

```
## [1] 0.1978935
```

We choose the lambda = 0.1978935, which is within 1 standard error of the minimum lambda as our best lambda.

Plot of the cross-validation error and the chosen lamda:

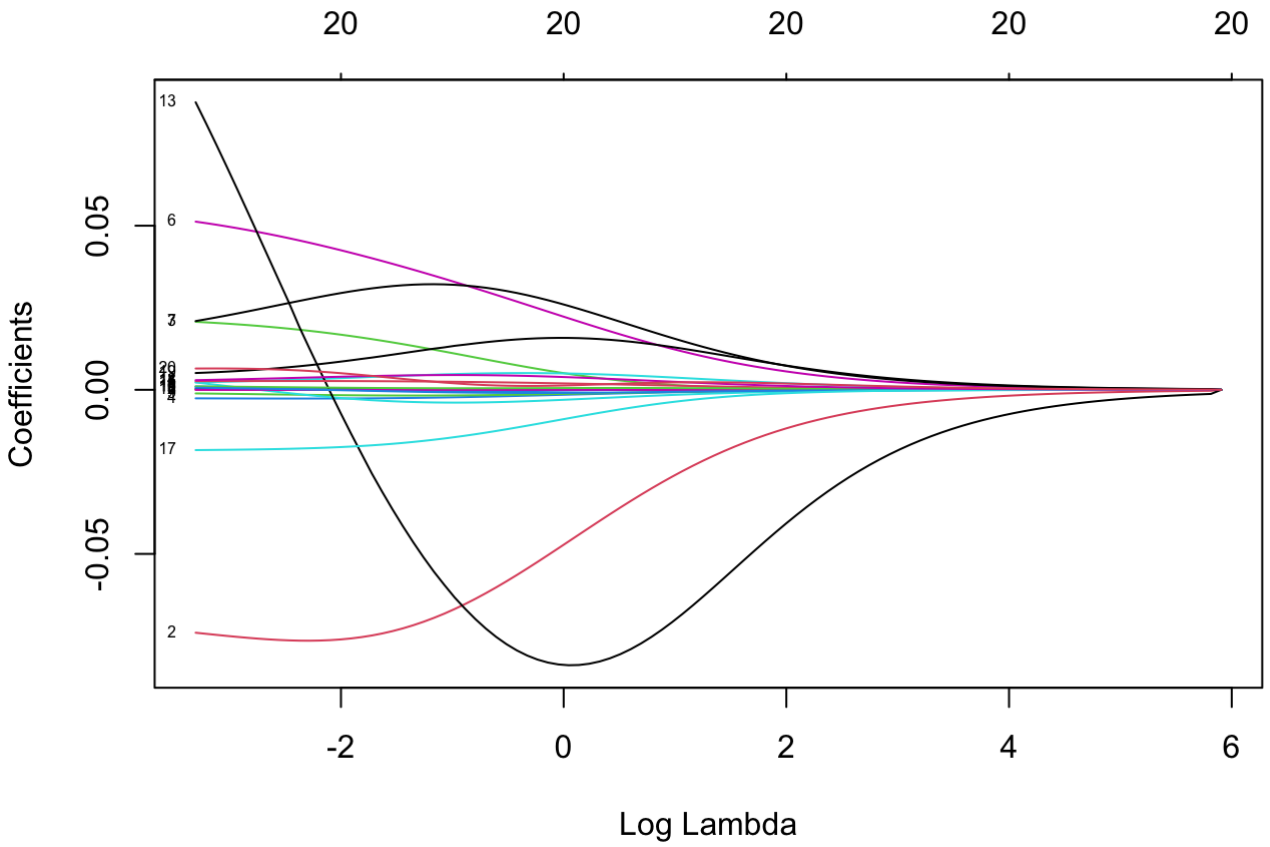
```
plot(cv.out_ridge)
abline(v=log(seleclam_ridge), col = "blue")
text(x= log(seleclam_ridge)+0.6, y=0.25, label="min MSE", cex=1)
```



We plot the MSE as a function of lambda. The number -1.7 on the top is the number of nonzero coefficient estimates.Confidence intervals represent error estimates for the loss metrics, which are the red dots. They are comupted using cross validation. The left vertical line shows the location of the minimum lambda and the right vertical line shows the location of the lambda which is within 1 standard error of the minimum lambda.

Plot of how the coefficients vary with lamda in a graph:

```
out_ridge = glmnet(x, y, alpha = 0) # Fit ridge regression model on the FULL dataset (tr
ain and test)

plot(out_ridge, xvar = "lambda",label = T)
```



In this graph, we can see all coefficients converges to zero as lambada larger than 3. When lambda smaller than 1, the coefficients has some large change as shown above.

ridge) The coefficients correspondent with the chosen lamda:

```
predict(out_ridge, type = "coefficients", s = seleclam_ridge)
```

```
## 21 x 1 sparse Matrix of class "dgCMatrix"
##                          1
## (Intercept)  4.522390e-01
## hours        1.989874e-04
## youngkids   -7.417525e-02
## oldkids      1.490201e-02
## age         -2.568806e-03
## education    4.057448e-03
## wage         3.916012e-02
## repwage      3.131802e-02
## hhours      -1.829603e-05
## hage        -1.731891e-03
## heducation  -2.182204e-04
## hwage       -3.354224e-03
## fincome     -9.082287e-07
## tax         -3.162295e-02
## meducation   2.617310e-03
## feducation   5.460734e-04
## unemp       -4.047487e-04
## cityyes     -1.667491e-02
## experience   4.269490e-03
## collegeyes   1.030334e-02
## hcollegeyes  3.907682e-03
```

```
ridge_pred = predict(ridge_mod, s = seleclam_ridge, newx = x_test)
# calculate MSE for test dataset
mean((ridge_pred - y_test)^2)
```

```
## [1] 0.08558048
```

Report the Error Rate:

```
#ridge model accuracy rate
ridge_pred = predict(ridge_mod, s = seleclam_ridge, newx = x)
ridge.pre <- as.vector(ifelse(ridge_pred >=0.5, 1, 0))
list1 <- ridge.pre== PSID1976$participation
sum(ifelse(list1 ==T,1,0))/753 # accuracy rate
```

```
## [1] 0.9309429
```

```
1-sum(ifelse(list1 ==T,1,0))/753
```

```
## [1] 0.0690571
```

We can see that the accuracy rate of ridge model is 0.9309429.

Lasso:

Tune the model: Use cross-validation to find the flexibly of the model (lamda).

```
lasso_mod = glmnet(x_train, y_train, alpha=1, lambda = grid)
set.seed(1)
cv.out_lasso = cv.glmnet(x_train, y_train, alpha = 1) # Fit ridge regression model on tr
aining data
cv.out_lasso
```

```
##
## Call:  cv.glmnet(x = x_train, y = y_train, alpha = 1)
##
## Measure: Mean-Squared Error
##
##      Lambda Measure       SE Nonzero
## min 0.01302 0.07894 0.004581      10
## 1se 0.05769 0.08331 0.004030       3
```

We use cross-validation to tune the model. The min value, 0.01302, is the lambda, which is the one which minimizes out-of-sample loss in CV. The 1se value, 0.05769, is the lambda, which is the largest lambda within 1 standard error of the minimum lambda.

The lamda which is within 1 standard error of the minimum lamda

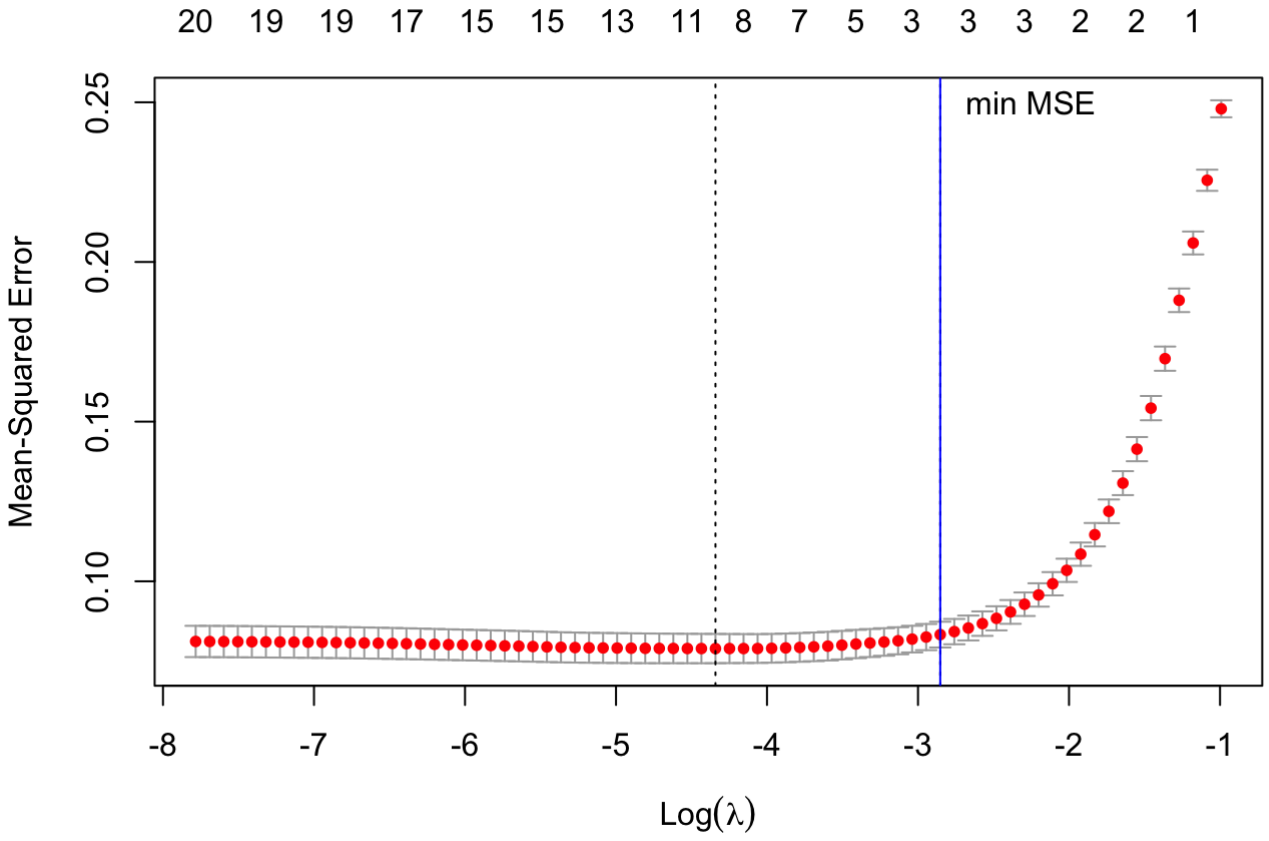
```
#log(bestlam)

seleclam_lasso = cv.out_lasso$lambda.1se  # Select lamda that minimizes training MSE, wh
ich also statisfy the within 1se of the min lambda
seleclam_lasso
```

```
## [1] 0.05768699
```

We choose the lambda = 0.05768699, which is within 1 standard error of the minimum lambda as our best lambda.

Show the cross-validation error and the chosen lamda in a graph:

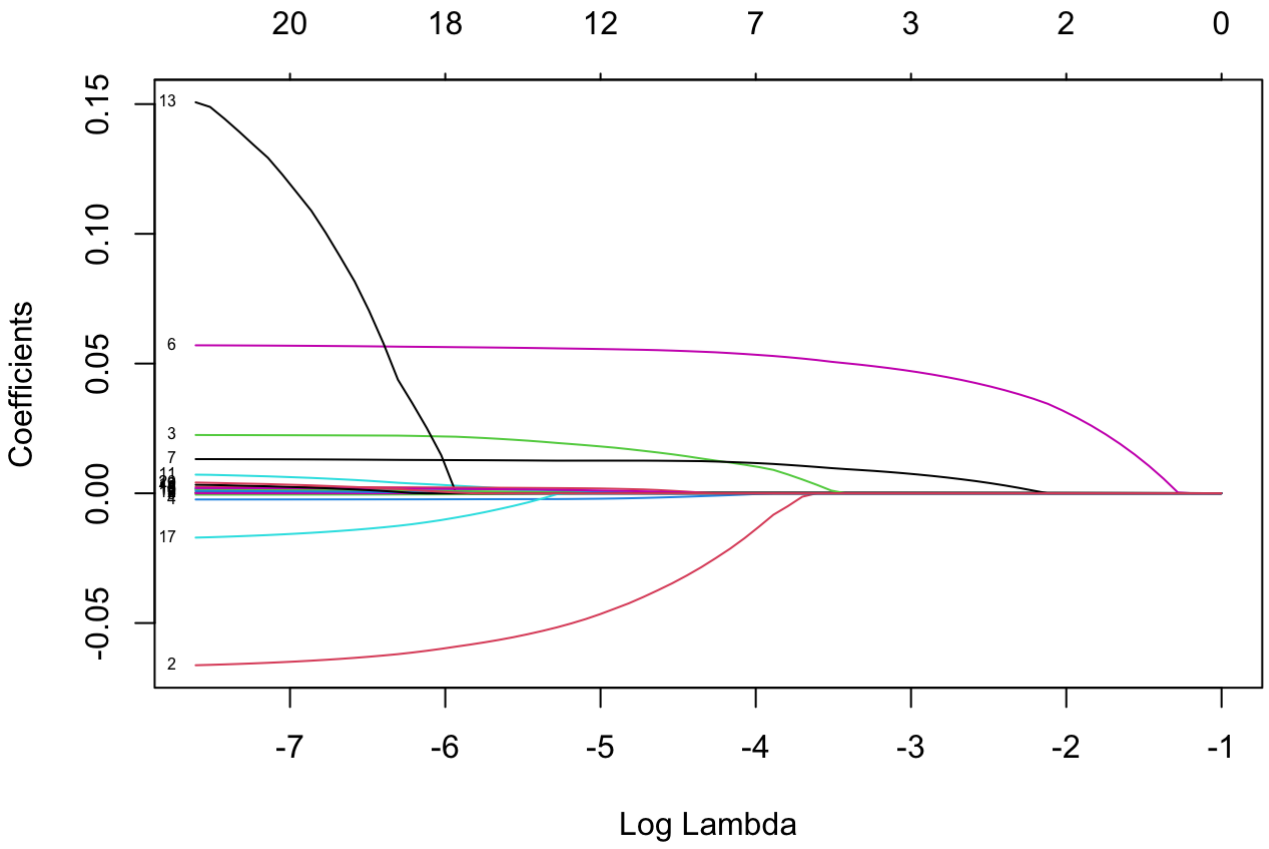
```
plot(cv.out_lasso) # show cross-validation error and the chosen lambda in a graph
abline(v=log(seleclam_lasso), col = "blue")
text(x= log(seleclam_lasso)+0.6, y=0.25, label="min MSE", cex=1)
```



We also plot the MSE as a function of lambda. The numbers on the top are the numbers of nonzero coefficient estimates. Confidence intervals represent error estimates for the loss metrics, which are the red dots. They are comupted using cross validation. The left vertical line shows the location of the minimum lambda and the right vertical line shows the location of the lambda which is within 1 standard error of the minimum lambda.

Show how the coefficients vary with lamda in a graph:

```
out_lasso = glmnet(x, y, alpha = 1) # Fit ridge regression model on the FULL dataset (tr
ain and test)

plot(out_lasso, xvar = "lambda",label = T)
```



Each colored line represents the value taken by a different coefficient in the model. As lambda grows, the regularization term has greater effect and there are fewer variables in the model because more and more coefficients will be 0.

Report the coefficients correspondent with the chosen lamda:

```
predict(out_lasso, type = "coefficients", s = seleclam_lasso)
```

```
## 21 x 1 sparse Matrix of class "dgCMatrix"
##                       1
## (Intercept) 0.246015674
## hours       0.000272287
## youngkids   .
## oldkids     .
## age         .
## education   .
## wage        0.045677247
## repwage     0.006630390
## hhours      .
## hage        .
## heducation  .
## hwage       .
## fincome     .
## tax         .
## meducation  .
## feducation  .
## unemp       .
## cityyes     .
## experience  .
## collegeyes  .
## hcollegeyes .
```

```
lasso_pred = predict(ridge_mod, s = seleclam_lasso, newx = x_test)
# calculate MSE for test dataset
mean((lasso_pred - y_test)^2)
```

```
## [1] 0.08001269
```

Report the Error Rate in the test subset:

```
lasso_pred = predict(lasso_mod, s = seleclam_lasso, newx = x)
lasso.pre <- as.vector(ifelse(lasso_pred >=0.5, 1, 0))
list1 <- lasso.pre== PSID1976$participation
sum(ifelse(list1 ==T,1,0))/753 # accuracy rate
```

```
## [1] 0.9163347
```

```
1-sum(ifelse(list1 ==T,1,0))/753
```

```
## [1] 0.08366534
```

We can see that the accuracy rate of lasso model is 0.9163347,

Compare the result Error rate:

Models	**Accurancy rate**	**Error rate**
Probit regression	0.7343958	0.2656042

Models	Accurancy rate	Error rate
Logistic regressions	0.7317397	0.2682603
KNN classification(k=25)	0.7869917	0.2130083
Lasso	0.9163347	0.08366534
Ridge	0.9309429	0.0690571

For all the models above, we can see that the ridge and lasso have a much better accuracy rate than the models we used in report2 and the ridge model has the highest accurancy rate and lowest error rate.

(extra) KNN classification

Firstly, we divide data to train and test subsets.

```
#install.packages("caret")
library(caret)
#is.factor(participation)
set.seed(1) #set seed.
index = createDataPartition(participation, p = 0.7, list = F )
train = PSID1976[index,]
test = PSID1976[-index,]
```

Then, we train our model on the training subset and choose the best K that minimizes the error rate in the test subset.

```
numbers = 10
repeats = 3
tunel = 20 # try 30 different K's
#x = trainControl(method = "repeatedcv",
                  #number = numbers,
                  #repeats = repeats,
                  #classProbs = TRUE,
                  #summaryFunction = twoClassSummary)
#model1 = train(as.factor(participation)~education+age+experience+tax+youngkids, data =
 train, method = "knn",
               #preProcess = c("center","scale"),
               #trControl = x,
               #metric = "ROC",
               #tuneLength = tunel)
# Summary of model
#model1
```

So, after predicting the results with on our entire data, we find the highest accuracy rate is 0.7869917 with k = 25.

Compared to the result with the Logistic regression, whose accuracy rate is 0.7317397, the KNN classification has a higher accuracy rate.

In general, KNN classification has the highest the highest accuracy rate, the Probit regression has the second highest accuracy rate, and the Logistic regressions has the lowest accuracy rate. However, their accuracy rates are vert close to each other, which is around 73% to 78%.

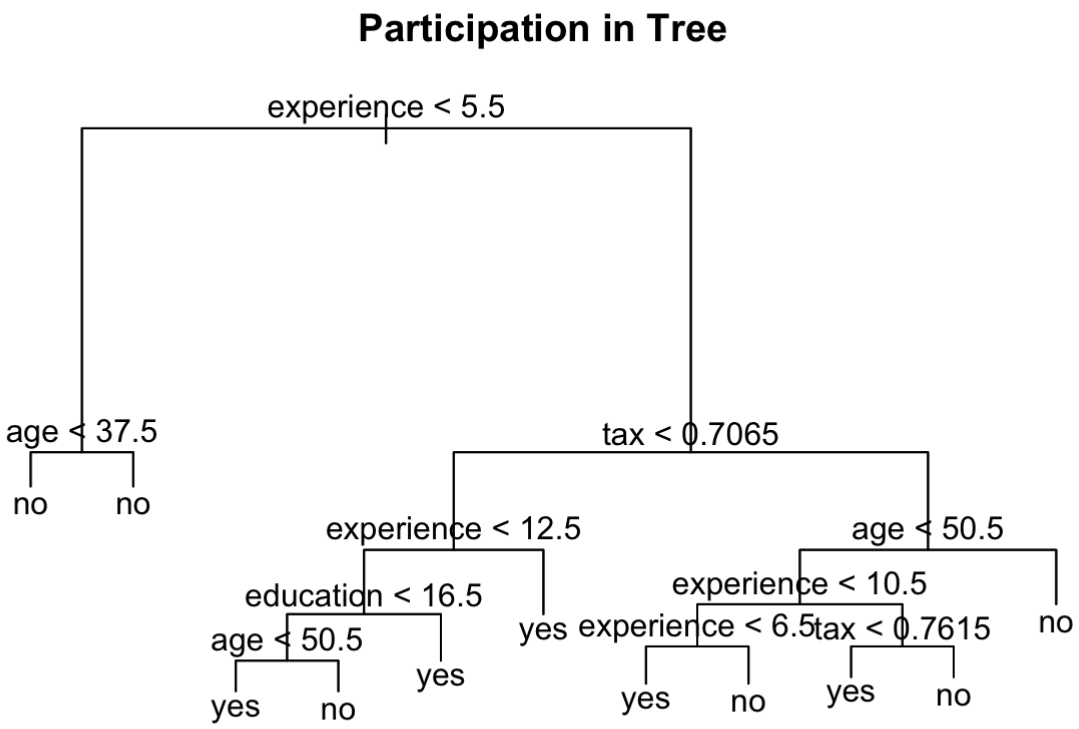
Decision tree

Fit and plot a big tree

```
train$participation <- as.factor(ifelse(train$participation ==1, "yes","no"))
test$participation <- as.factor(ifelse(test$participation ==1, "yes","no"))
tree1 = tree(participation~education+age+experience+tax+youngkids, train)
summary(tree1)
```

```
##
## Classification tree:
## tree(formula = participation ~ education + age + experience +
##     tax + youngkids, data = train)
## Variables actually used in tree construction:
## [1] "experience" "age"        "tax"        "education"
## Number of terminal nodes:  11
## Residual mean deviance:  1.041 = 538 / 517
## Misclassification error rate: 0.2348 = 124 / 528
```

```
plot(tree1) #plot the tree
text(tree1, pretty = 0)
title("Participation in Tree")
```



Check the error rate on the test subset:

```
tree_full_pred = predict(tree1, test, type = "class")
mean(tree_full_pred==y_test)
```

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

```
error_rate_tree_full = 1- mean(tree_full_pred==y_test)
error_rate_tree_full
```

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

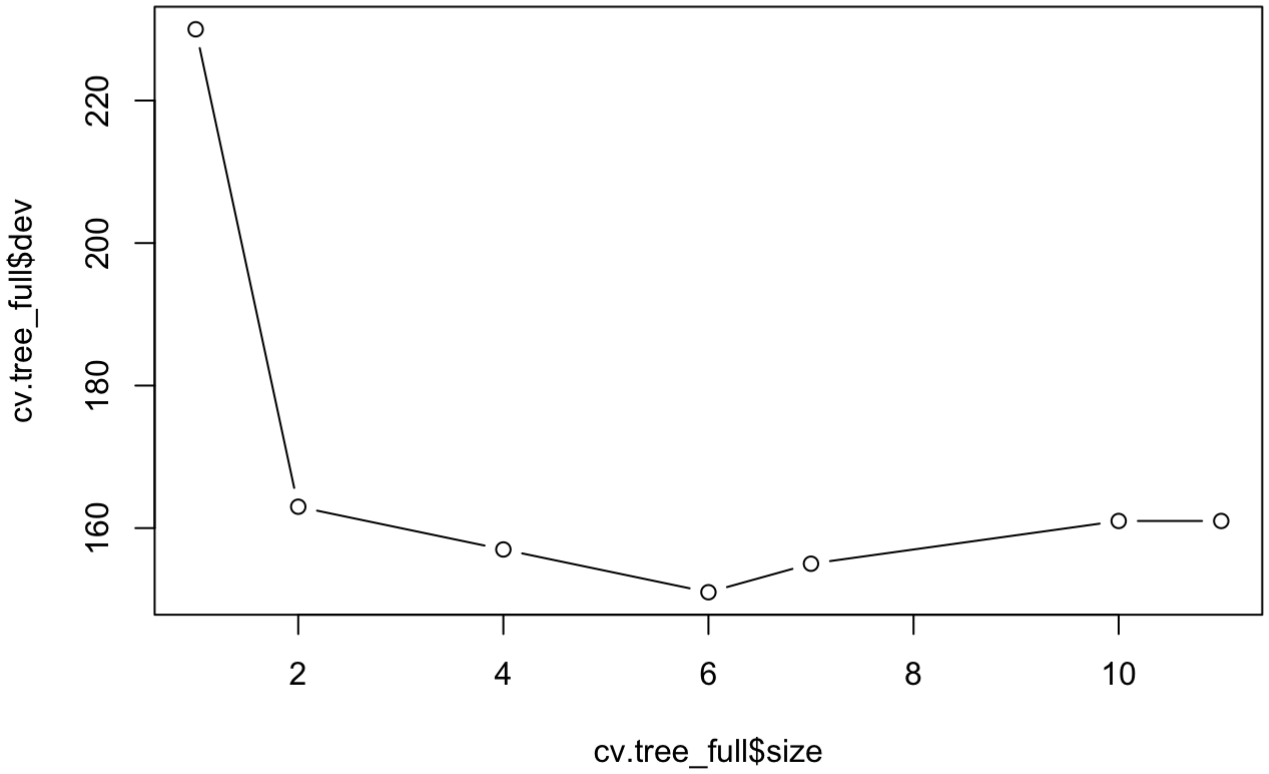
The error rate of the full tree is 0.

Use cross validation to prune your tree:

```
cv.tree_full = cv.tree(tree1, FUN = prune.misclass)
cv.tree_full
```

```
## $size
## [1] 11 10  7  6  4  2  1
##
## $dev
## [1] 161 161 155 151 157 163 230
##
## $k
## [1]      -Inf  0.000000  1.666667  3.000000  5.500000  7.500000 70.000000
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```
plot(cv.tree_full$size, cv.tree_full$dev, type = "b")
```

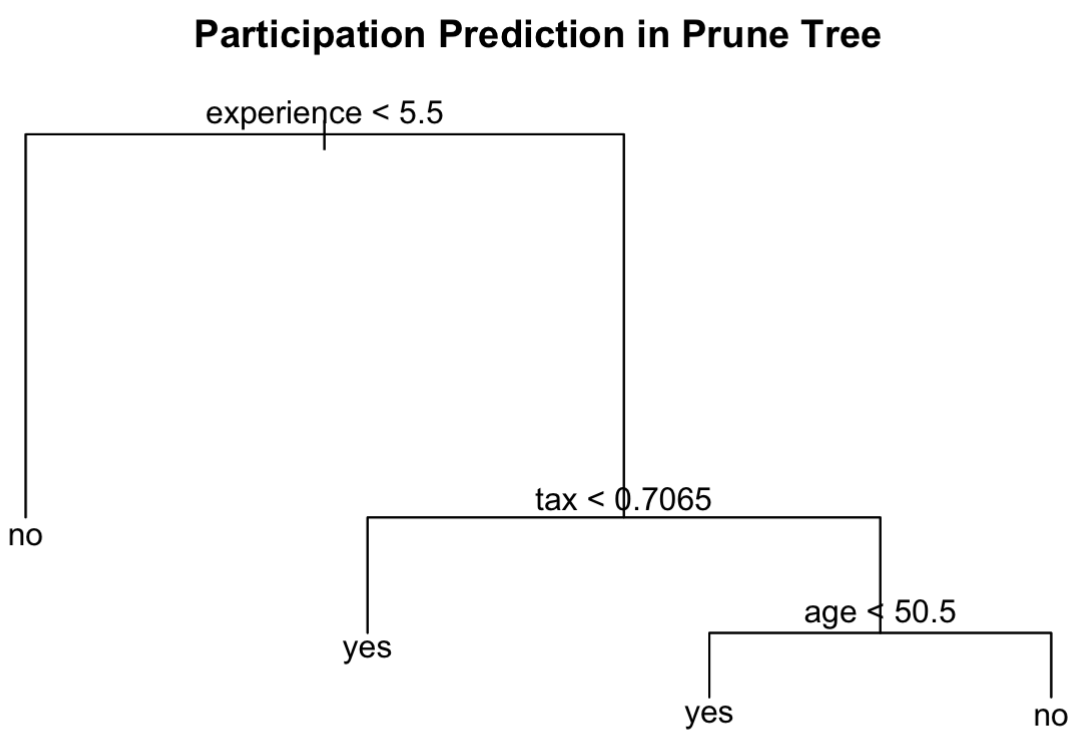


We see from this plot that the tree with 4 terminal nodes results in the lowest cross-validation error rate, with 164 cross-validation errors.

We now apply the prune.misclass() function in order to prune the tree to obtain the nine-node tree by setting the parameter best = 4, we choose 4 becuase it is the simpliest tree:

Plot the pruned tree:

```
tree_prune = prune.misclass(tree1, best = 4)
plot(tree_prune)
text(tree_prune, pretty = 0)
title("Participation Prediction in Prune Tree")
```



Interpret your tree (the selected variables and splits):

If the experience is less than 5.5 years, the participation results would be no. For those who have experience larger than 5.5 years, if age is less than 46.5, the participation results would be yes. For those who have experience larger than 5.5 years and the age is larger than 46.5 and tax is larger than 0.6765, the participation results would be no.

```
tree_prune_pred = predict(tree_prune, test, type = "class")
mean(tree_prune_pred==y_test)
```

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

```
error_rate_tree_prune = 1- mean(tree_prune_pred==y_test)
error_rate_tree_prune
```

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

Models	Accurancy rate	Error rate
Probit regression	0.7343958	0.2656042
Logistic regressions	0.7317397	0.2682603
KNN classification(k=25)	0.7869917	0.2130083

Models	Accurancy rate	Error rate
Lasso	0.9377778	0.0622222
Ridge	0.9111111	0.0888889
Full Tree	0.6888889	0.3111111
Prune Tree	0.6622222	0.3377778

As the chart shown above, the prediction error rate for the tree models are larger than other models, and the prune tree will also increase the error rate from the full tree. So the tree models dosen't perform well in this dataset.

Boot-strap

Boot 100 trees. Report the test subseterror rate with this prediction

```
arlist <- NA
for (i in c(1:100)){
set.seed(i)
PSID_train = PSID1976 %>%
  sample_frac(.7)

PSID_test = PSID1976 %>%
  setdiff(PSID_train)
tree_PSID=tree(participation~., PSID_train)
prune_PSID = prune.tree(tree_PSID,
                        best = 2)
single_tree_estimate = predict(prune_PSID,
                               newdata = PSID_test)

tree.pre <- ifelse(single_tree_estimate==1, "yes","no")

list5 <- as.vector(tree.pre==test$participation)
ar <- sum(ifelse(list5 ==T,1,0))/226 # accuracy rate

arlist[i] <- ar
}

arlist
```

```
##   [1] 0.9292035 0.9911504 0.9646018 0.9911504 0.9690265 0.9734513 0.9646018
##   [8] 0.9734513 0.9734513 0.9867257 0.9646018 0.9778761 0.9424779 0.9911504
##  [15] 0.9690265 0.9292035 0.9823009 0.9690265 0.9955752 0.9690265 0.9911504
##  [22] 0.9601770 0.9911504 0.9823009 0.9955752 0.9778761 0.9911504 0.9734513
##  [29] 0.9424779 0.9557522 0.9823009 0.9557522 0.9823009 0.9823009 0.9955752
##  [36] 0.9690265 0.9734513 0.9778761 0.9690265 0.9336283 0.9557522 0.9336283
##  [43] 0.9778761 0.9690265 0.9646018 0.9601770 0.9557522 0.9557522 0.9823009
##  [50] 0.9955752 0.9513274 0.9734513 0.9955752 0.9955752 0.9911504 0.9424779
##  [57] 0.9734513 0.9867257 0.9734513 0.9646018 0.9911504 0.9469027 0.9646018
##  [64] 0.9911504 0.9601770 0.9823009 0.9292035 0.9690265 0.9513274 0.9823009
##  [71] 0.9513274 0.9690265 0.9867257 0.9380531 0.9380531 0.9778761 0.9911504
##  [78] 0.9734513 0.9601770 0.9646018 0.9513274 0.9867257 0.9778761 0.9823009
##  [85] 0.9601770 0.9646018 0.9778761 0.9911504 0.9469027 0.9823009 0.9867257
##  [92] 0.9823009 0.9601770 0.9867257 0.9469027 0.9867257 0.9867257 0.9823009
##  [99] 0.9823009 0.9557522
```

```
mean(arlist)
```

```
## [1] 0.9708407
```

Compare the boot-strap aggregated model performance with the performance the single tree. The mean of the error rate of the boot-strap aggregated model is the same as the single tree, with a small std error of 0.02385018.

The LASSO and Ridge models can largely decrease the error rate, and the tree models will increase the error rate. So, the LASSO and Ridge models are the best models so far to predict the participation of labor market in 1976.

Split to train dataset and test dataset. The train dataset is for training and fitting, and we will use the test dataset to test the accuracy rate.

Bagging classification

Bagging is simply a special case of a random forest with m = p. Therefore, the randomForest() function can be used to perform both random forests and bagging. We are going to set the number of variables $mtry$ to the number of all the characteristics in the PSID1976 data set.

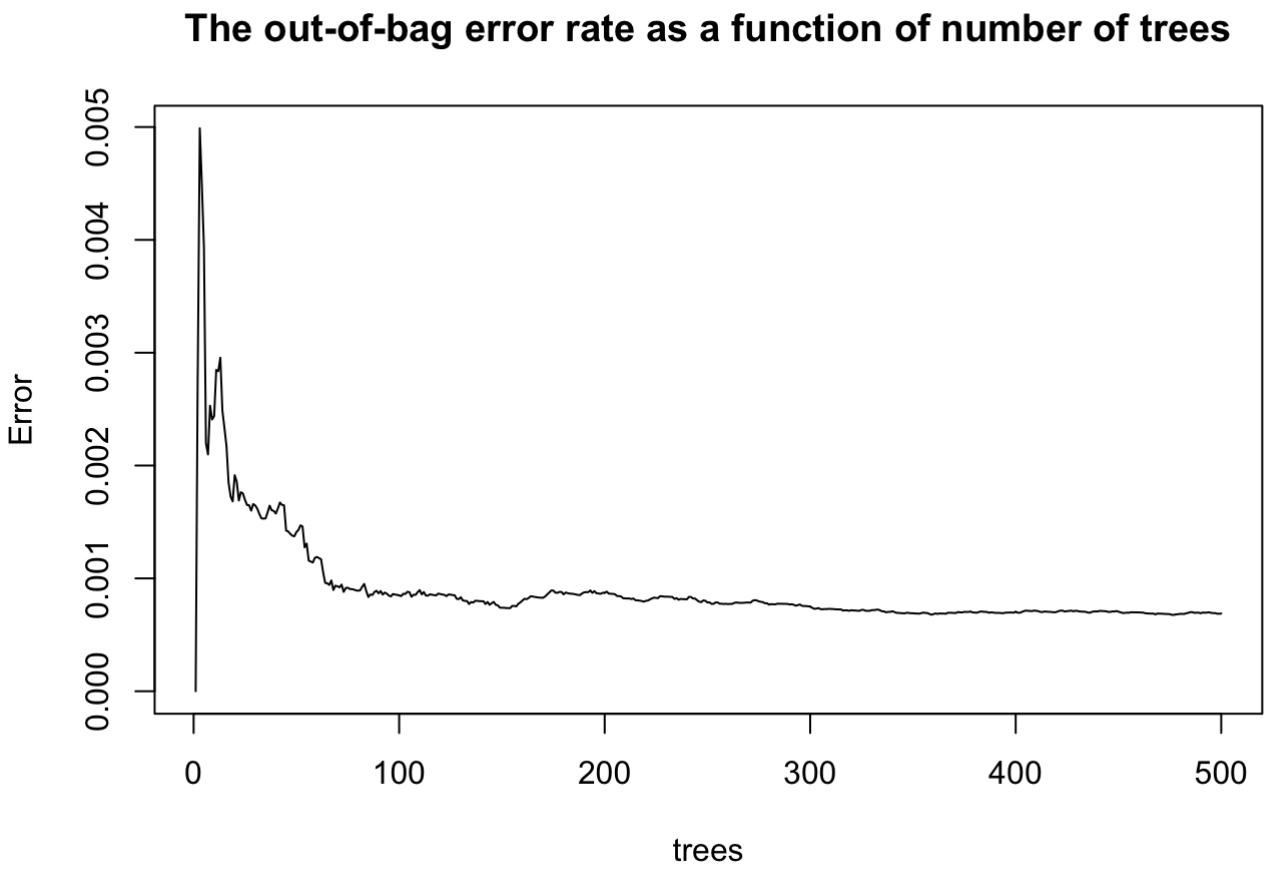
Plot the out-of-bag error rate as a function of number of trees

```
library(MASS)
library(randomForest)
library(dplyr)
library(ggplot2)
library(AER)
library(glmnet)
library(tidyverse)
library(class)
data("PSID1976")
PSID1976$participation <- ifelse(PSID1976$participation=='yes',1,0)

set.seed(23)
data_train = PSID1976 %>%
  sample_frac(.8)
data_test = PSID1976 %>%
  setdiff(data_train)
bag.data = randomForest(participation ~., data=data_train,
                          mtry=ncol(data_train)-1,
                          importance=TRUE)
bag.data
```

```
##
## Call:
##  randomForest(formula = participation ~ ., data = data_train,      mtry = ncol(data_t
rain) - 1, importance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 20
##
##           Mean of squared residuals: 0.0006885596
##                     % Var explained: 99.72
```

The argument mtry=ncol(boston_train)-1 = 20 indicates that all 20 predictors should be considered for each split of the tree.

```
plot(bag.data, main = "The out-of-bag error rate as a function of number of trees")
```



In the plot, we can see the the out-of-bag error rate as a function of number of trees. The error rate will largely increase at the begining and largely decrease. And as the number of trees is more than around 70, the error rate are almost the same and lower than 0.001. Plot predicted Y vs actual Y if you have a regression and error rate table if you have a classification

```
yhat.bag = predict(bag.data,
                           newdata = data_test)

yhat.bag[yhat.bag>.5]="yes"
yhat.bag[yhat.bag<=.5]="no"
table(yhat.bag,data_test$participation)
```

```
##
## yhat.bag  0  1
##       no  64  0
##       yes  0 87
```

We can see from the error rate table that the model successfully predict all the data points in our data and the error rate is 0 and the accuracy rate is 1.

Compare the error rate to that in Lasso/Ridge models in your last report:

Models	Accurancy rate	Error rate
Lasso	0.9377778	0.0622222

Models	Accurancy rate	Error rate
Ridge	0.9111111	0.0888889
Bagging classification	1	0

By comparing with Lasso/Ridge models, we can see in the above chart that. the error rate of the Bagging classification is the smallest, which is equal to 0. And then it is Lasso and Ridge. And all three models have very low error rates. The Bagging classification preform the best.

Plot the importance matrix:

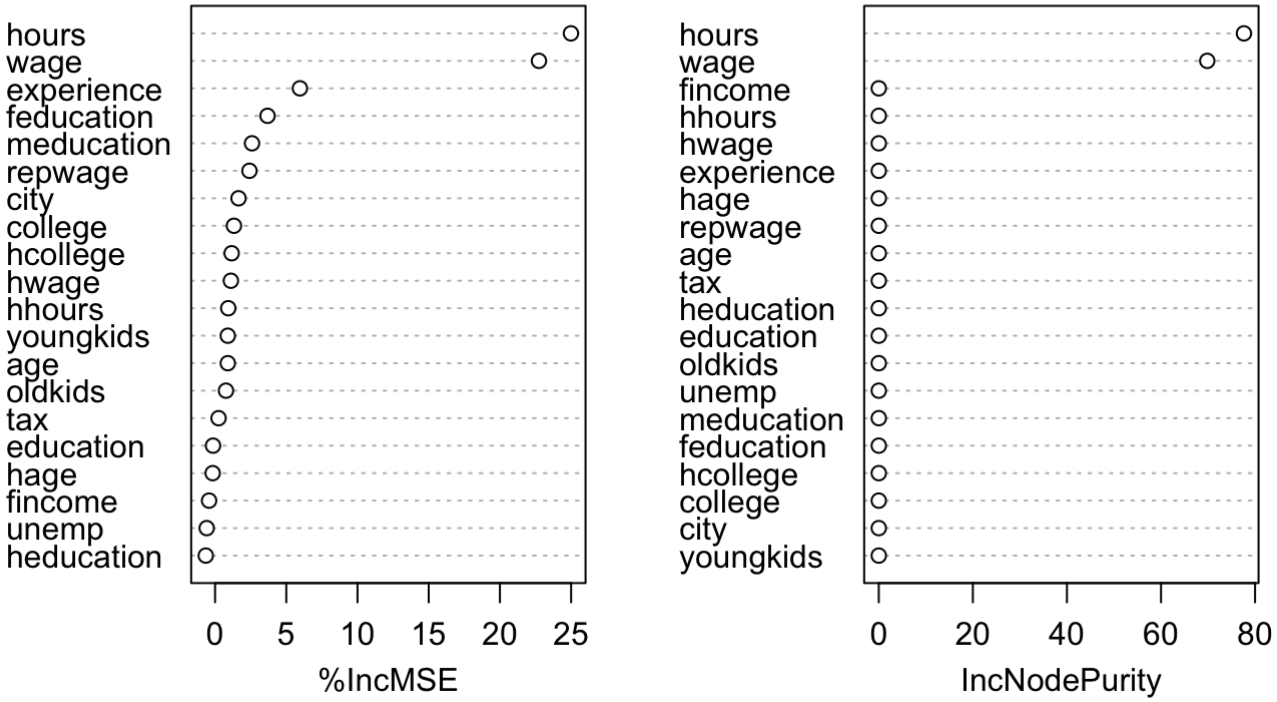
```
importance(bag.data)
```

```
##                %IncMSE IncNodePurity
## hours       24.9872784  7.763270e+01
## youngkids    0.8939844  2.943867e-15
## oldkids      0.7673181  2.899103e-14
## age          0.8938844  5.006395e-14
## education   -0.1430149  3.898348e-14
## wage        22.7395026  6.983160e+01
## repwage      2.4139226  5.037304e-14
## hhours       0.9254306  6.926459e-14
## hage        -0.1676992  5.065814e-14
## heducation  -0.6518350  4.310552e-14
## hwage        1.1112108  6.299361e-14
## fincome     -0.4202451  7.561773e-14
## tax          0.2462096  4.526690e-14
## meducation   2.5945384  1.558798e-14
## feducation   3.6824762  1.529354e-14
## unemp       -0.5869459  2.493028e-14
## city         1.6465039  1.186251e-14
## experience   5.9560600  5.807754e-14
## college      1.3247160  1.347900e-14
## hcollege     1.1641647  1.468692e-14
```

Two measures of variable importance are reported. The former is based upon the mean decrease of accuracy in predictions on the out-of-bag samples when a given variable is excluded from the model. The latter is a measure of the total increase in node impurity that results from splits over that variable, averaged over all trees. If we drop the $hours$ variable, the mean decrease of accuracy in predictions on the out-of-bag samples will be 2.378246%. If we drop the $wage$ variable, the mean decrease of accuracy in predictions on the out-of-bag samples will be 2.260392%.

```
varImpPlot(bag.data)
```

bag.data



After visualize the importance matrix, the results indicate that across all of the trees considered in the bagging classificayion, the hours and the wages are by far the two most important variables.

Random Forest classification

Growing a random forest proceeds in exactly the same way, except that we use a smaller value of the mtry argument, only a subset of variables are used at each split. By default, randomForest() uses p/3 variables when building a random forest of regression trees, and √p variables when building a random forest of classification trees. This helps de-correlate the trees.

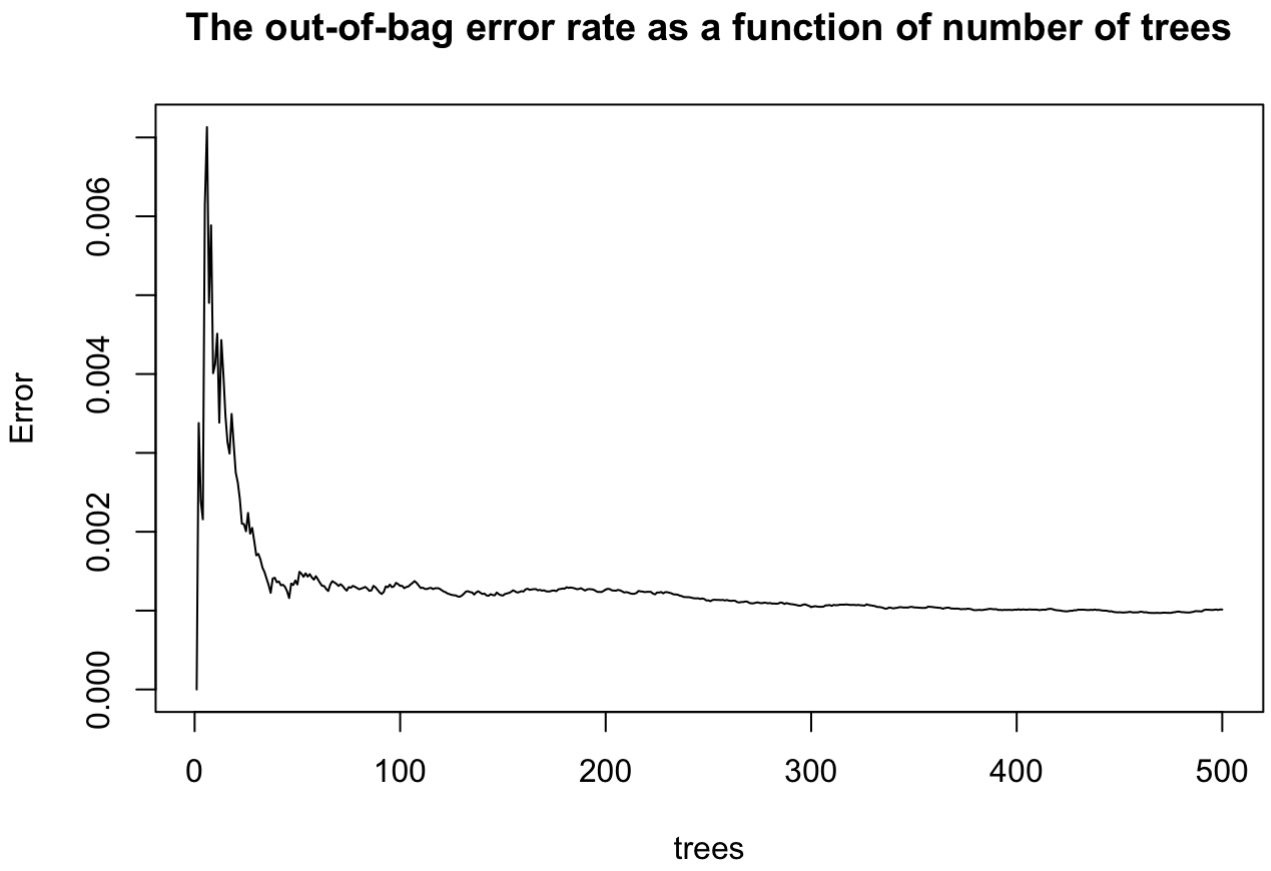
Plot the out-of-bag error rate as a function of number of predictors considered in each split

Here we use mtry = 5

```
set.seed(23)

rf.data = randomForest(participation~.,
                       data = data_train,
                       mtry = 5,
                       importance = TRUE,
                       do.trace = 100)
```

```
##      |      Out-of-bag    |
## Tree |      MSE  %Var(y) |
##  100 | 0.001315     0.54 |
##  200 | 0.001266     0.52 |
##  300 | 0.001046     0.43 |
##  400 | 0.001011     0.41 |
##  500 | 0.001014     0.41 |
```

```
plot(rf.data, main = "The out-of-bag error rate as a function of number of trees")
```



In this plot of the error rate as a function of number of trees, we can see that the error rate decrease gradually and after the number of trees go after 150, the error rate stay at about 0.002.

Use the out of bag error to tune the number of predictors in each split (mtry):

Now, instead of using random number for the mtry around square root of the total number of variables, we need to find the best number of variables to use by estimating the Out Of Bag (OOB) error. I use hyperparameter tuning by looping over mtry and comparing the error rate for each value

```
oob.err<-double(20)
test.err<-double(20)

#mtry is no of Variables randomly chosen at each split
for(mtry in 1:20)
{
  rf=randomForest(participation~.,
                          data = data_train,, mtry=mtry, ntree=400)
  oob.err[mtry] = rf$mse[400] #Error of all Trees fitted on training

  pred<-predict(rf,data_test) #Predictions on Test Set for each Tree
  test.err[mtry]= with(data_test, mean( (participation - pred)^2)) # "Test" Mean Squared
Error
}
```

```
round(test.err ,2)
```

```
##  [1] 0.04 0.01 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
## [16] 0.00 0.00 0.00 0.00 0.00
```

```
round(oob.err,2)
```

```
##  [1] 0.04 0.01 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
## [16] 0.00 0.00 0.00 0.00 0.00
```

```
which.min(round(oob.err,2))
```

```
## [1] 3
```

So, mtry = 3 would be the best mtry. We redo the random forest with mtry = 3.

```
rf.data = randomForest(participation~.,
                          data = data_train,
                          mtry = 3,
                          importance = TRUE,
                          do.trace = 100)
```

```
##      |      Out-of-bag   |
## Tree |      MSE  %Var(y) |
##  100 | 0.003617     1.47 |
##  200 | 0.002942     1.20 |
##  300 | 0.002613     1.06 |
##  400 | 0.002677     1.09 |
##  500 |  0.00272     1.11 |
```

Show the error rate table:

```
yhat.rf = predict(rf.data, newdata = data_test)
yhat.rf[yhat.rf>.5]="yes"
yhat.rf[yhat.rf<=.5]="no"
table(yhat.rf,data_test$participation)
```

```
##
## yhat.rf  0  1
##      no  64  0
##      yes  0 87
```

We can see from the error rate table that the model successfully predict all the data points in our data and the error rate is 0 and the accuracy rate is 1.

Compare the error ratein Lasso/Ridge models with other models

Models	**Accurancy rate**	**Error rate**
Lasso	0.9377778	0.0622222
Ridge	0.9111111	0.0888889
Bagging classification	1	0
Random Forest classification	1	0

So the error rate of the Bagging classification and Random Forest classification are the smallest, both are 0. And then it is Lasso and Ridge. And all four models have very low error rates. The Bagging classification and Random Forest classification models perform the best.

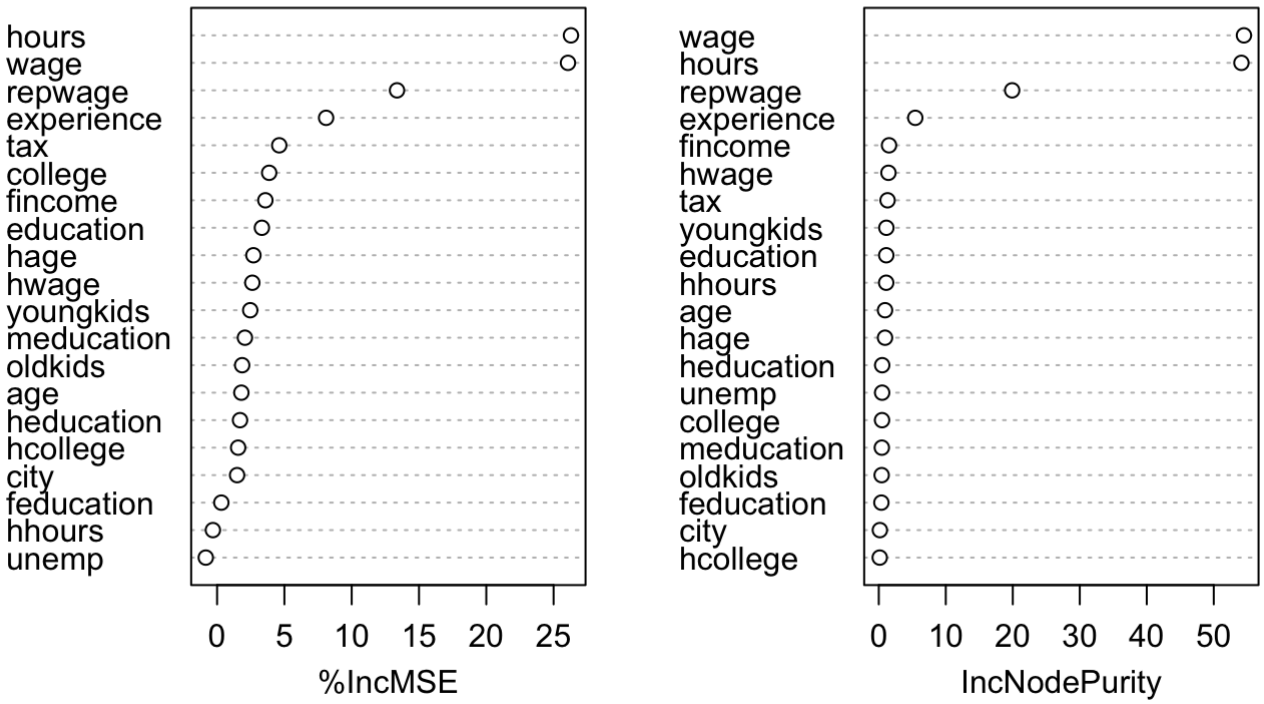
Plot the importance matrix:

```
importance(rf.data)
```

```
##                %IncMSE IncNodePurity
## hours       26.2889989    54.1295639
## youngkids    2.4619712     1.1300077
## oldkids      1.8623157     0.4291758
## age          1.8030739     0.9344389
## education    3.3286728     1.1154000
## wage        26.0675479    54.5069126
## repwage     13.3666771    19.9085637
## hhours      -0.3114613     1.1050089
## hage         2.6995054     0.9325561
## heducation   1.7040840     0.5090636
## hwage        2.6182596     1.4430522
## fincome      3.5876918     1.5299919
## tax          4.6166644     1.3113475
## meducation   2.0679094     0.4557993
## feducation   0.3151335     0.3869948
## unemp       -0.8396935     0.5012949
## city         1.4860854     0.1622974
## experience   8.0991531     5.4460387
## college      3.8688717     0.4882163
## hcollege     1.5638887     0.1407927
```

Again, two measures of variable importance are reported. The former is based upon the mean decrease of accuracy in predictions on the out-of-bag samples when a given variable is excluded from the model. The latter is a measure of the total increase in node impurity that results from splits over that variable, averaged over all trees. If we drop the $hours$ variable, the mean decrease of accuracy in predictions on the out-of-bag samples will be 24.8796816%. If we drop the $wage$ variable, the mean decrease of accuracy in predictions on the out-of-bag samples will be 25.6533090%.

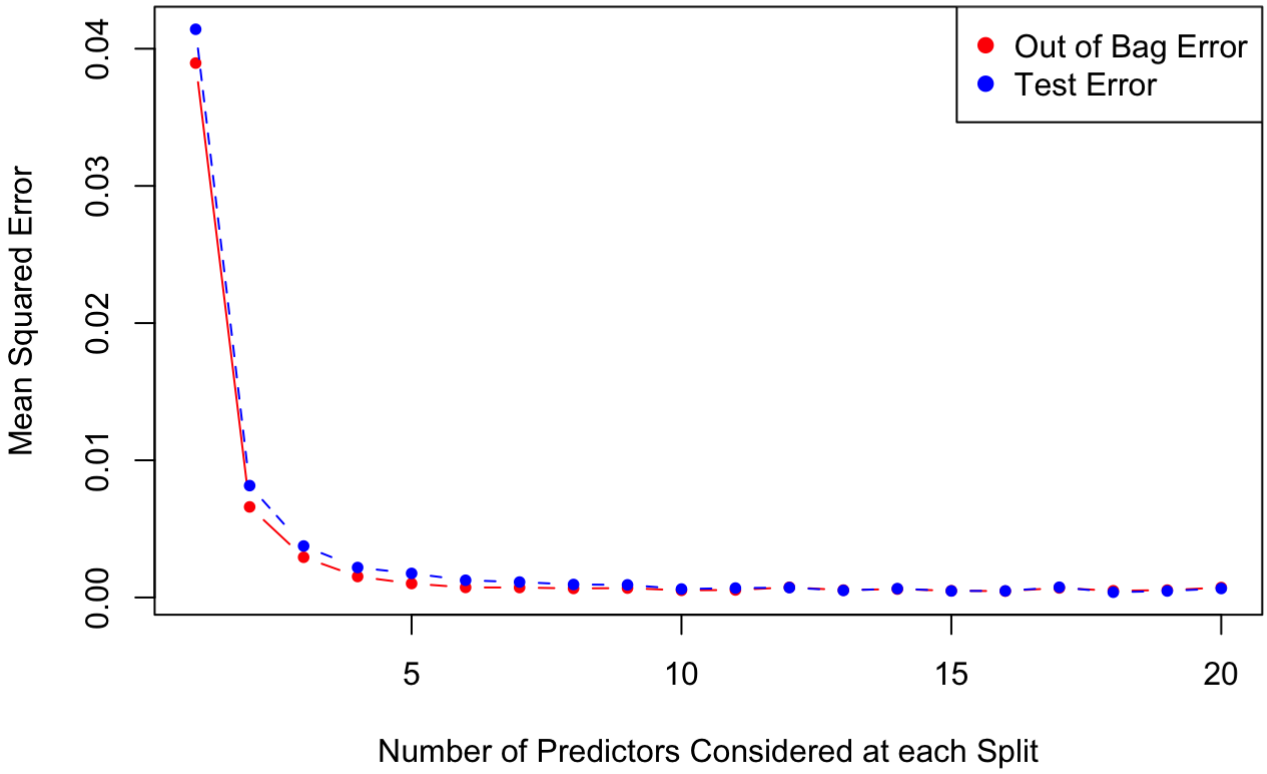
```
varImpPlot(rf.data)
```



After visualize the importance matrix, the results indicate that across all of the trees considered in the bagging classificayion, the hours and the wages are still by far the two most important variables, and we can see the repwage is the third most important variable.

Plot the test error and out-of-bag error in a same graph vs mtry and show that they follow a similar pattern:

```
matplot(1:mtry , cbind(oob.err,test.err), pch=20 , col=c("red","blue"),type="b",ylab="Me
an Squared Error",xlab="Number of Predictors Considered at each Split")
legend("topright",legend=c("Out of Bag Error","Test Error"),pch=19, col=c("red","blue"))
```



By plotting these two error rates to visualize the best number of variables, we can see that the test error and out-of-bag error vs mtry are two lines exactly the same. So, they do follow a similar pattern.

Boosting classification

In boosting, the trees are trained sequentially based on the reside of the previous step. Unlike random forests and bagging, boosting can overfit if the number of trees (B) is very large. An appropriate choice of B can be made using cross-validation. The number of splits d controls the tree complexity.

Now we fit the boosting classification using the parameters in the lab. Plot error rate table if you have a classification:

```
boost.data = gbm(participation~.,
                   data = data_train,
                   distribution = "bernoulli",
                   n.trees = 5000,
                   interaction.depth = 4)
yhat.boost = predict(boost.data,
                         newdata = data_test,
                         n.trees = 5000)

yhat.boost[yhat.boost>.5]="yes"
yhat.boost[yhat.boost<=.5]="no"
table(yhat.boost,data_test$participation)
```

```
##
## yhat.boost  0  1
##         no  64  0
##         yes  0 87
```

We can see from the error rate table that the model successfully predict all the data points in our data and the error rate is 0 and the accuracy rate is 1.

Compare the test error rateto the Lasso/Ridge/Bagging/RandomForest models:

Models	**Accurancy rate**	**Error rate**
Lasso	0.9377778	0.0622222
Ridge	0.9111111	0.0888889
Bagging classification	1	0
Random Forest classification	1	0
Boosting classification	1	0

So the error rate of the Boosting classification, Bagging classification and Random Forest classification are the smallest, both are 0. And then it is Lasso and Ridge. And all five models have very low error rates.The Boosting classification, Bagging classification and Random Forest classification models perform the best.

Plot the importance matrix:

The summary() function produces a relative influence plot and also outputs the relative influence statistics:

```
#summary(boost.data)
```

Again, we see that wage and hours are again the most important variables by far. We can also produce partial dependence plots for these two variables.

XGboost classification

XGBoost, or eXtreme Gradient Boosting, implements gradient boosting, but now includes a regularization parameter and implements parallel processing. It also has a built-in routine to handle missing values. XGBoost also allows one to use the model trained on the last iteration, and updates it when new data becomes available.

We need to put our data into matrix format.

```
train <- as.data.frame(PSID1976[index,])
test <- as.data.frame(PSID1976[-index,])


x = model.matrix(participation~., PSID1976)[,-1]
y = PSID1976$participation

x_train = model.matrix(participation~., train)[,-1]
y_train = train$participation

x_test = model.matrix(participation~., test)[,-1]
y_test = test$participation
dtrain <- xgb.DMatrix(data = x_train, label = y_train)
```

```
data.xgb = xgboost(data=dtrain,
                     max_depth=2,
                     eta = 0.1,
                     nrounds=40, # max number of boosting iterations (trees)
                     lambda=0,
                     print_every_n = 10,
                     objective="binary:logistic") # for classification: objective = "bin
ary:logistic"
```

```
## [1]  train-error:0.000000
## [11] train-error:0.000000
## [21] train-error:0.000000
## [31] train-error:0.000000
## [40] train-error:0.000000
```

```
yhat.xgb <- predict(data.xgb,x_test)
table(yhat.xgb,y_test)
```

```
##                      y_test
## yhat.xgb                0   1
##   0.00883977301418781  97   0
##   0.991160213947296     0 128
```

We can see from the error rate table that the model successfully predict all the data points in our data and the error rate is 0 and the accuracy rate is 1.

Compare the test error rate to the Lasso/Ridge/Bagging/RandomForest/Boosting models

Models	**Accurancy rate**	**Error rate**
Lasso	0.9377778	0.0622222
Ridge	0.9111111	0.0888889
Bagging classification	1	0
Random Forest classification	1	0

Models	Accurancy rate	Error rate
Boosting classification	1	0
XGboost classification	1	0

So the error rate of the Boosting classification, Bagging classification， XGboost classification and Random Forest classification are the smallest, both are 0. And then it is Lasso and Ridge. And all six models have very low error rates. And Boosting classification, Bagging classification， XGboost classification and Random Forest classification perform the best.

Plot the importance matrix

```
importance <- xgb.importance(colnames(x_train),model=data.xgb)
importance
```

```
##      Feature         Gain       Cover  Frequency
## 1:     hours 1.000000e+00 0.873737864 0.85483871
## 2: youngkids 6.649527e-09 0.043657717 0.04838710
## 3:       age 5.115021e-09 0.028440580 0.03225806
## 4:   oldkids 5.115021e-09 0.046248061 0.04838710
## 5:    hhours 5.115021e-10 0.007915779 0.01612903
```

```
xgb.plot.importance(importance, rel_to_first=TRUE, xlab="Relative Importance")
```

In this plot, we can see that the $hours$ is the most important variable.

(extra) Tune parameters using grid search for the Boosting model

We will tuning parameters in Boosting (learning rate, number of trees, number of splits in each tree)

```
# create hyperparameter grid
hyper_grid <- expand.grid(
shrinkage = c(0.01, 0.05, 0.1),
interaction.depth = c(3, 4, 5))



# grid search
for(i in 1:nrow(hyper_grid)) {
set.seed(233)
# train model
gbm.tune <- gbm(
  formula = participation~.,
  distribution = "bernoulli",
  data = data_train,
  n.trees = 5000,
  interaction.depth = hyper_grid$interaction.depth[i],
  shrinkage = hyper_grid$shrinkage[i],
  train.fraction = 0.75,
  verbose = FALSE
  )

# add min training error and trees to grid
hyper_grid$optimal_trees[i] <- which.min(gbm.tune$valid.error)
hyper_grid$min_error[i] <- min(gbm.tune$valid.error)
}

hyper_grid %>%
dplyr::arrange(min_error) %>%
head(5)
```

```
##   shrinkage interaction.depth optimal_trees    min_error
## 1      0.10                 3          2478 8.940604e-16
## 2      0.10                 4          2262 2.587481e-14
## 3      0.10                 5          2396 1.749791e-11
## 4      0.05                 3          2210 1.645614e-09
## 5      0.05                 5          4990 1.868081e-09
```

By runing a grid search, I find the best boosting model with n.trees = 73, interaction.depth = 3, shrinkage = 0.1.

```
boost.data.ec = gbm(participation~.,
                    data = data_train,
                    distribution = "bernoulli",
                    n.trees = 73,
                    shrinkage = 0.1,
                    interaction.depth = 3)
best_boost_pred = predict(boost.data.ec, data_test, n.trees=73)

best_boost_pred[best_boost_pred>.5]="yes"
best_boost_pred[best_boost_pred<=.5]="no"
table(best_boost_pred,data_test$participation)
```

```
##
## best_boost_pred  0  1
##              no  64  0
##              yes  0 87
```

We can see from the error rate table that the model successfully predict all the data points in our data and the error rate is 0 and the accuracy rate is 1.

Compare the test MSE (error rate) to the Lasso/Ridge/Bagging/RandomForest/Boosting models

Models	**Accurancy rate**	**Error rate**
Lasso	0.9377778	0.0622222
Ridge	0.9111111	0.0888889
Bagging classification	1	0
Random Forest classification	1	0
Boosting classification	1	0
XGboost classification	1	0
Boosting classification after tuning	1	0

Since the error rate for Boosting classification is 0, there is no space for improvement after tuning. Both of the models have the error rate of 0. So the error rate of the Boosting classification, Bagging classification， Boosting classification after tuning, XGboost classification and Random Forest classification are the smallest, both are 0. And then it is Lasso and Ridge. And all models have very low error rates.

Plot the importance matrix

```
#summary(boost.data.ec)
```

Again, we can see that the wage is the most important variable.

Surprisingly, all of those models predict our dataset with 0 error rate, which makes it hard to analysis which one is the best among them. However, we can see that those models perform better than the models in our previous reports, such as Lasso, Ridge,Probit,Logistic， KNN. Also, we can see that the wages and hours all the most important variables in the dataset.

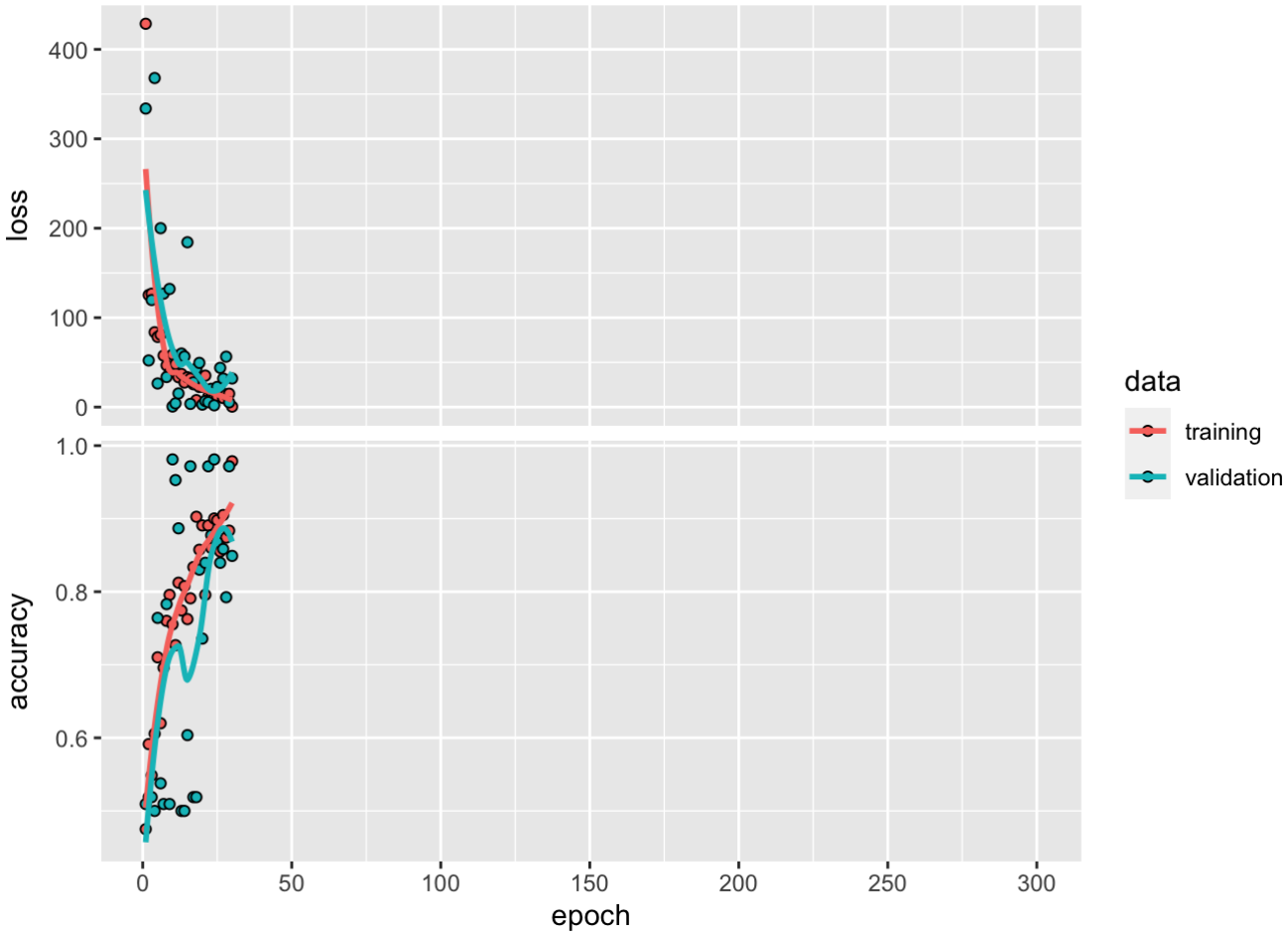
Neural Net

```
library(keras)
library(ISLR)
library(dplyr)
library(tensorflow)
#install.packages("keras")
library(keras)
#install_keras()
#install_tensorflow()
set.seed(233)

#PSID1976$participation <- ifelse(PSID1976$participation=='yes',1,0)
PSID1976$city <- ifelse(PSID1976$city=='yes',1,0)
PSID1976$college <- ifelse(PSID1976$college=='yes',1,0)
PSID1976$hcollege <- ifelse(PSID1976$hcollege=='yes',1,0)

train_nn = PSID1976 %>%
  sample_frac(.7)
test_nn = PSID1976 %>%
  setdiff(train_nn)
```

```
train_labels <- to_categorical(train_nn[,"participation"],2)
train_data <- as.matrix(train_nn[!names(train_nn) %in% c("participation")])

test_data <- as.matrix(test_nn[!names(train_nn) %in% c("participation")])
test_labels <- to_categorical(test_nn[,"participation"],2)
```

```
model <- keras_model_sequential() %>%
  layer_dense(units = 64, activation = "relu",
              input_shape = dim(train_data)[2]) %>%
  layer_dense(units = 64, activation = "relu") %>%
  layer_dense(units = 64, activation = "relu") %>%
  layer_dense(units = 2, activation= "softmax")

model %>% compile(
  loss = 'categorical_crossentropy',
  optimizer = optimizer_rmsprop(),
  metrics = c('accuracy')
)
early_stop <- callback_early_stopping(monitor = "val_loss", patience = 20)

epochs=300
history_class <- model %>% fit(
  train_data,
  train_labels,
  epochs = epochs,
  validation_split = 0.2,
  callbacks = list(early_stop)
)
plot(history_class)
```

```
## `geom_smooth()` using formula 'y ~ x'
```



From this plot, we can see that the accurancy rate of Neural Net will increases after epochs and finally get to around 96%.

(extra)SVM

We can perform cross-validation using tune() to select the best choice of γ and cost for an SVM with a radial kernel

```
library(dplyr)
library(ggplot2)
library(e1071)
tune.out = tune(svm, participation~., data = train_nn, kernel = "radial",
                ranges = list(cost = c(0.1,1,10,100,1000), gamma = c(0.5,1,2,3,4)))
```

```
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##    10   0.5
##
## - best performance: 0.1991675
##
## - Detailed performance results:
##     cost gamma     error  dispersion
## 1  1e-01   0.5 0.3606555 0.046129577
## 2  1e+00   0.5 0.1993682 0.006616666
## 3  1e+01   0.5 0.1991675 0.006492421
## 4  1e+02   0.5 0.1991675 0.006492421
## 5  1e+03   0.5 0.1991675 0.006492421
## 6  1e-01   1.0 0.3745319 0.047583938
## 7  1e+00   1.0 0.2387039 0.005499755
## 8  1e+01   1.0 0.2384381 0.005101934
## 9  1e+02   1.0 0.2384381 0.005101934
## 10 1e+03   1.0 0.2384381 0.005101934
## 11 1e-01   2.0 0.3763169 0.047936145
## 12 1e+00   2.0 0.2474243 0.005947472
## 13 1e+01   2.0 0.2471682 0.005402852
## 14 1e+02   2.0 0.2471682 0.005402852
## 15 1e+03   2.0 0.2471682 0.005402852
## 16 1e-01   3.0 0.3764534 0.047989989
## 17 1e+00   3.0 0.2482447 0.006224308
## 18 1e+01   3.0 0.2479776 0.005640076
## 19 1e+02   3.0 0.2479776 0.005640076
## 20 1e+03   3.0 0.2479776 0.005640076
## 21 1e-01   4.0 0.3764712 0.047999166
## 22 1e+00   4.0 0.2483569 0.006279884
## 23 1e+01   4.0 0.2480866 0.005688418
## 24 1e+02   4.0 0.2480866 0.005688418
## 25 1e+03   4.0 0.2480866 0.005688418
```

```
bestmod = tune.out$best.model
```

```
summary(bestmod)
```

```
##
## Call:
## best.tune(method = svm, train.x = participation ~ ., data = train_nn,
##     ranges = list(cost = c(0.1, 1, 10, 100, 1000), gamma = c(0.5,
##         1, 2, 3, 4)), kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  10
##       gamma:  0.5
##     epsilon:  0.1
##
##
## Number of Support Vectors:  525
```

Therefore, the best choice of parameters involves cost = 10 and gamma = 0.5. We can plot the resulting fit using the plot() function, and view the test set predictions for this model by applying the predict() function to the test data.

```
plot(bestmod, train_nn)
```

```
pred = predict(tune.out$best.model, newdata = test_nn)
pred[pred>.5]="yes"
pred[pred<=.5]="no"
table(pred,test_nn$participation)
```

```
##
## pred    0   1
##   no   32   3
##   yes  56 135
```

The accurancy is 0.845

Comparing All Models

Models	Accurancy rate
Logistic(education+age+tax+experience+youngkids)	0.7317397;
Logistic(education+age+experience+youngkids)	0.7383798;
Logistic(age+experience+youngkids)	0.7144754;
Logistic(age+experience)	0.6653386;
Logistic(experience)	0.6706507;
Logistic(age)	0.5710491;
Logistic(youngkids)	0.622842;

Models	Accurancy rate
Logistic(education)	0.5909695;
Logistic(tax)	0.5962815;
Probit Classification	0.7343958
Lasso	0.9377778
Ridge	0.9111111
FULL Decision tree	0.6888889
Prune Tree	0.6622222
Boot-strap	0.9543805
Bagging classification	1
Random Forest classification	1
Boosting classification	1
XGboost classification	1
Boosting classification after tuning	1
Neural Net	0.96
KNN	0.7869917
SVM	0.845
From the chart above, we can see that Bagging classifica	tion, Random Forest classification, Boosting classification, and XGboost classification will have the highest accuranct rate for 100% and neural net, lasso, ridge, boot-strap also have high accurancy rate around 95%.

Conclusion

Firstly, in terms of the correletion between each variable and participation, we can see that education and experience will positively affect the participation, and tax, number of young kids, will negatively effects the partcipantion. Secondly, from the Logistic Classificatio, we can see that all the varibales will affect the participantion, so the full model with more varibles would work the best and have the highest accurancy rate. And expeirence would be the most determinant variable. According to the p-value of each variable, education, age, experience, youngkids have p-values lower than 0.001, so they are significantly different from zero, with confidence level larger than 99%. And tax would have the highest standard error. Thirdly, the accuracy of Probit regression and Logistic regression are very close. Probit regression's accuracy rate is slightly higher than the Logistic regression's accuracy rate, both of them are around 73%. Fourthly, for KNN model, with K = 25, we have the highest accuracy rate, around 78%. Fifthly, for ridge and Lasso, we can see that the accuracy rate of ridge model is 0.9309429, and the accuracy rate of lasso model is 0.9163347. Both of them are largely increase the accurancy rate of the models above. Sixthly, the decision tree, 2e see from this plot that the tree with 4 terminal nodes results in the lowest cross-validation error rate, with 164 cross-validation errors. The full tree will also

increase the error rate from the full tree, with accurancy rate around 68%. Seventhym, the error rate of the Boosting classification, Bagging classification， XGboost classification and Random Forest classification are the smallest, both are 0. And all the models have very low error rates. And Boosting classification, Bagging classification， XGboost classification and Random Forest classification perform the best. Lastly, in the Neural Net model we use, the accurancy rate is around 96%. Although this might not be the best Neural Net that fits our dataset, the accurancy rate is farily high. In generel, Bagging classification, Random Forest classification, Boosting classification, and XGboost classification will have the highest accuranct rate for 100% and neural net, lasso, ridge, boot-strap also have high accurancy rate around 95%.logistic model and tree models have a lower rate of prediction for this dataset.