ECON882 Assignment 1

Mehtab Hanzroh

2025-02-19

ECON882 Assignment 1

Importing Libraries

Loading Data

```
setwd("/media/mehtab/6266-3261/Documents/PhD/TA/ECON882 Winter 2024")
df = read.csv("men2015b.csv")
```

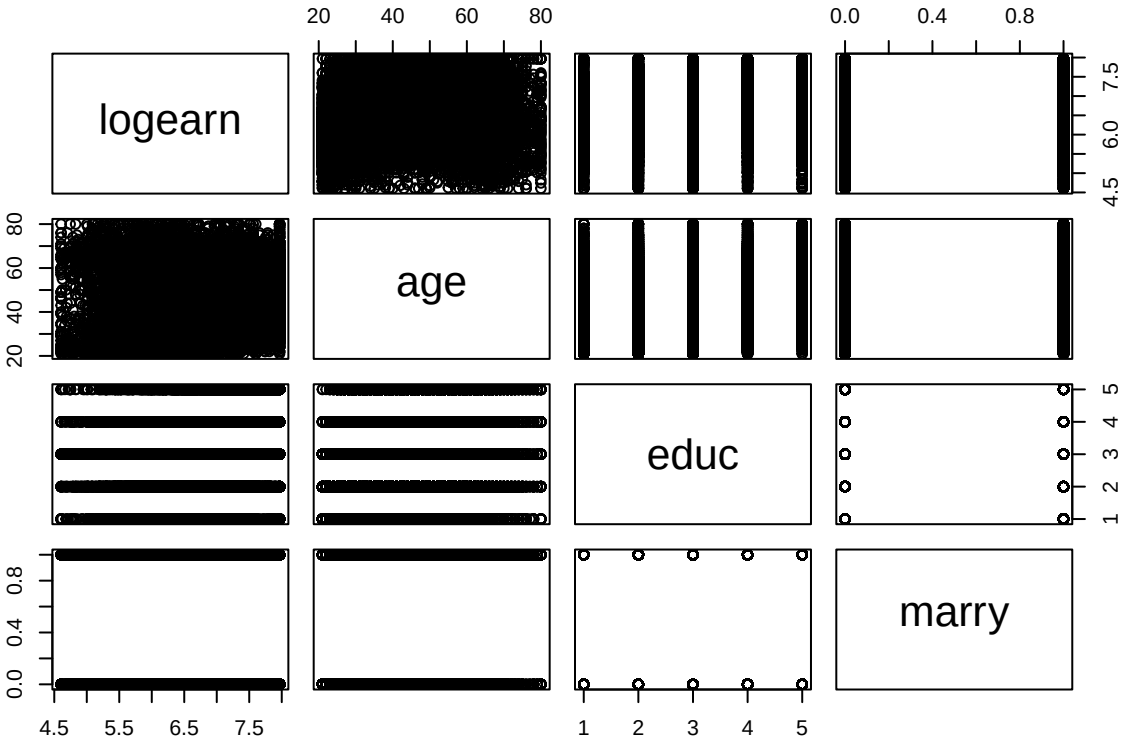
Question 1

First we will create the log(earnings) variable.

```
df$logearn = log(df$earnings)
```

Let's start by looking at the pairs plot.

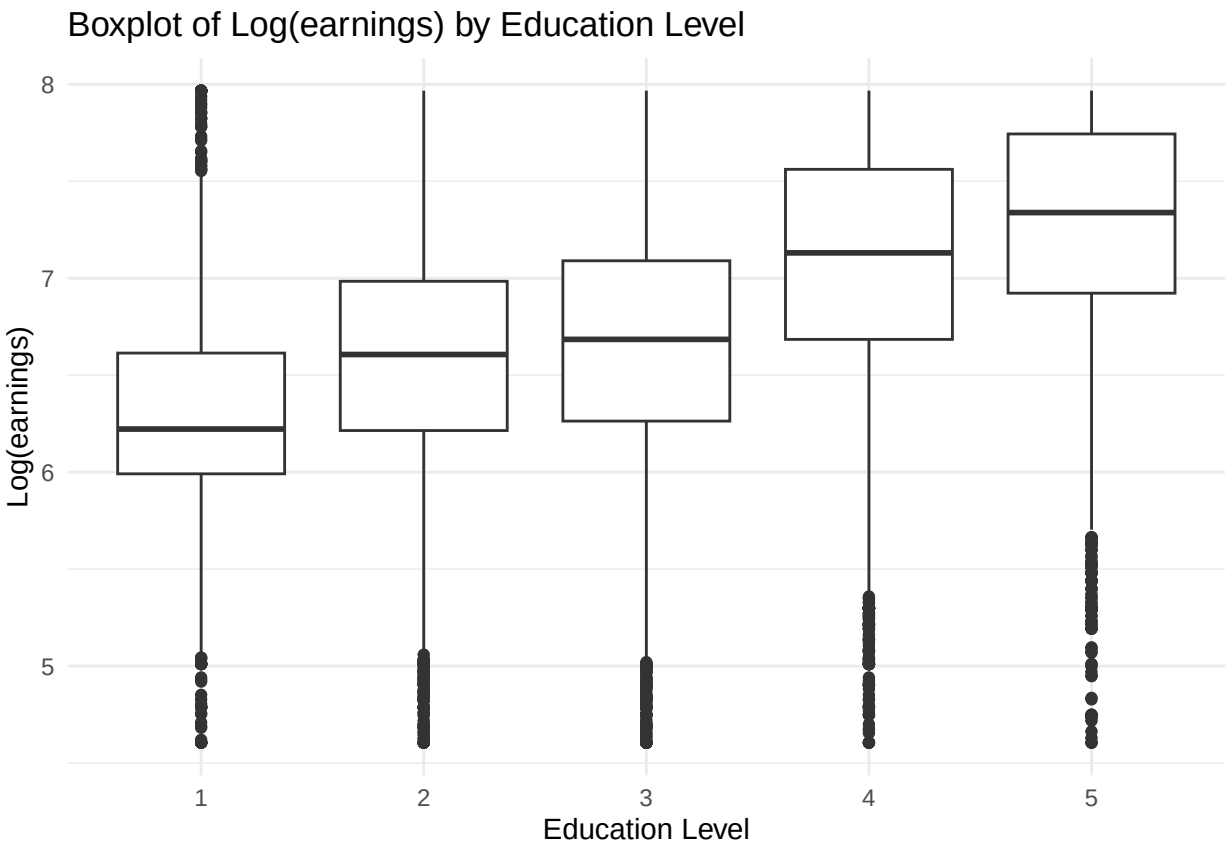
```
pairs(df[,c("logearn","age","educ","marry")])
```



"marry" is a dummy variable, so we will include this in the model as is. "educ" is a categorical variable, so we

will include dummy variables for each of its levels (except for one of-course). Including it on its own would be a mistake, since it assumes the effect on earnings from each jump in education levels is identical. This is easily verified graphically by plotting the distribution (boxplot) of logearn for each level of educ. This is simple in ggplot.

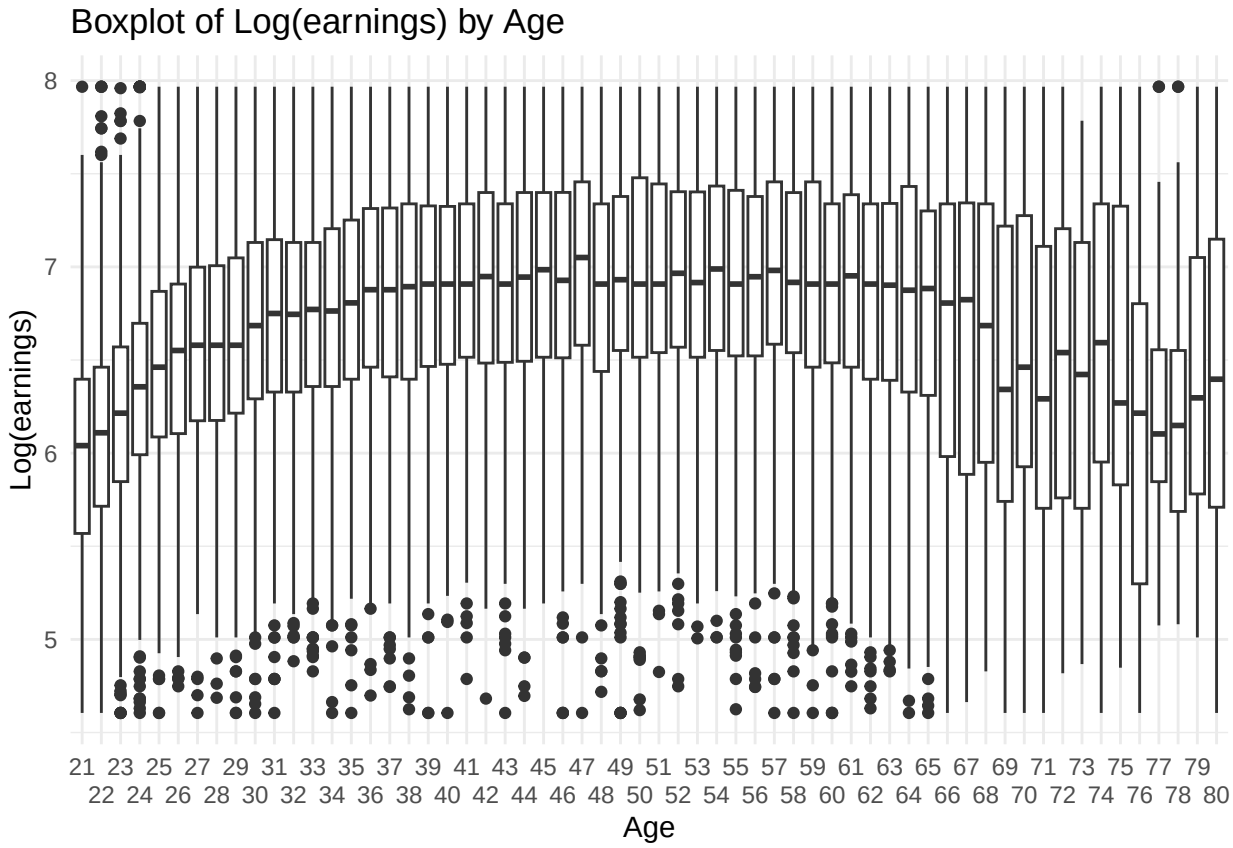
```
(ggplot(df, aes(x=factor(educ),y=logearn)) +
    geom_boxplot() +
    theme_minimal() +
    labs(x='Education Level',y='Log(earnings)',title='Boxplot of Log(earnings) by Education Level'))
```



Note. It is the fact that we surrounded educ by factor() that generates a separate boxplot for each level of educ.

The relation is clearly non-linear. The relation between logearn and age is hard to see from the pairs plot, so let's create the same plot for the mean of logearn by age.

```
(ggplot(df, aes(x=factor(age),y=logearn)) +
    geom_boxplot() +
    theme_minimal() +
    labs(x='Age',y='Log(earnings)',title='Boxplot of Log(earnings) by Age') +
    scale_x_discrete(guide = guide_axis(n.dodge = 2)))
```



A quadratic relation between logearn and age seems appropriate. Let's estimate the relevant linear regression model.

```
model = lm(logearn ~ age + I(age^2) + factor(educ) + marry, data=df)
summary(model)
```

```
##
## Call:
## lm(formula = logearn ~ age + I(age^2) + factor(educ) + marry,
##     data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.73317 -0.34904  0.02994  0.40200  1.94108
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)    4.395e+00  3.574e-02  122.95   <2e-16 ***
## age            8.154e-02  1.620e-03   50.34   <2e-16 ***
## I(age^2)      -8.473e-04  1.771e-05  -47.84   <2e-16 ***
## factor(educ)2  3.235e-01  1.265e-02   25.56   <2e-16 ***
## factor(educ)3  3.973e-01  1.274e-02   31.19   <2e-16 ***
## factor(educ)4  7.691e-01  1.302e-02   59.07   <2e-16 ***
## factor(educ)5  9.279e-01  1.449e-02   64.04   <2e-16 ***
## marry          1.544e-01  7.194e-03   21.46   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 0.5838 on 32429 degrees of freedom
## Multiple R-squared:  0.2711, Adjusted R-squared:  0.271
## F-statistic:  1723 on 7 and 32429 DF,  p-value: < 2.2e-16
```

All variables are highly significant. A cubic term for age may be appropriate as well, so let's see if it is important for the regression fit.

```
model_cubic = lm(logearn ~ age + I(age^2) + I(age^3) + factor(educ) + marry, data=df)
summary(model_cubic)
```

```
##
## Call:
## lm(formula = logearn ~ age + I(age^2) + I(age^3) + factor(educ) +
##     marry, data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.72342 -0.34880  0.03149  0.40188  1.88752
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)    4.158e+00  9.927e-02   41.88  < 2e-16 ***
## age            9.890e-02  6.974e-03   14.18  < 2e-16 ***
## I(age^2)      -1.241e-03  1.547e-04   -8.02 1.09e-15 ***
## I(age^3)       2.799e-06  1.094e-06    2.56   0.0105 *
## factor(educ)2  3.247e-01  1.266e-02   25.64  < 2e-16 ***
## factor(educ)3  3.986e-01  1.274e-02   31.27  < 2e-16 ***
## factor(educ)4  7.695e-01  1.302e-02   59.10  < 2e-16 ***
## factor(educ)5  9.274e-01  1.449e-02   64.00  < 2e-16 ***
## marry          1.524e-01  7.235e-03   21.06  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5838 on 32428 degrees of freedom
## Multiple R-squared:  0.2713, Adjusted R-squared:  0.2711
## F-statistic:  1509 on 8 and 32428 DF,  p-value: < 2.2e-16
```

While the coefficient on the squared term does change (note we did not use orthogonal polynomials), the fit hardly changes. We will thus proceed without the cubic term, to be parsimonious and avoid over-fitting concerns. Some of you may also test using interactions, but in my testing interactions do not improve fit very much, and raise over-fitting concerns.

Question 2

To get these predictions, we need to create a dataframe with the relevant data. Since the hypothetical individuals have different combinations for each possible values of age, educ, and marital status, this is simple with `expand.grid()`.

```
hyp_df = expand.grid(age=c(30,40,50,60,70), educ=c(4,5), marry=c(0,1))
hyp_df = cbind(hyp_df,predict(model,hyp_df,level=.99,interval="prediction"))
hyp_df
```

```
##    age educ marry      fit      lwr      upr
## 1   30    4     0 6.847432 5.343341 8.351524
## 2   40    4     0 7.069697 5.565585 8.573809
## 3   50    4     0 7.122501 5.618366 8.626637
## 4   60    4     0 7.005845 5.501686 8.510003
```

```
## 5   70    4     0 6.719728 5.215383 8.224073
## 6   30    5     0 7.006263 5.502064 8.510463
## 7   40    5     0 7.228528 5.724318 8.732738
## 8   50    5     0 7.281332 5.777109 8.785555
## 9   60    5     0 7.164676 5.660439 8.668913
## 10  70    5     0 6.878559 5.374144 8.382973
## 11  30    4     1 7.001807 5.497709 8.505905
## 12  40    4     1 7.224072 5.720002 8.728142
## 13  50    4     1 7.276876 5.772808 8.780944
## 14  60    4     1 7.160220 5.656132 8.664308
## 15  70    4     1 6.874103 5.369808 8.378397
## 16  30    5     1 7.160638 5.656443 8.664833
## 17  40    5     1 7.382903 5.878745 8.887060
## 18  50    5     1 7.435707 5.931562 8.939852
## 19  60    5     1 7.319050 5.814895 8.823206
## 20  70    5     1 7.032933 5.528580 8.537287
```

Earnings are maximized for individual 18 in the dataframe, a 50 year old, with the highest level of education and who is married. This is not surprising given the graphs above.

Question 3

We will estimate with `knn.reg()`. caret does not implement the efficient LOOCV algorithm for KNN discussed in class, and thus using it will take a prohibitively long time.

```
ks = seq(1,100,1)
r_sq = c()
for (i in ks){
    r2pred = knn.reg(train=df[,c('age','educ','marry')],y=df$logearn,k=i)$R2Pred
    r_sq = c(r_sq,r2pred)
}
optk = ks[which.max(r_sq)]
paste0('The maximum LOO Predictive R-squared is ',round(max(r_sq),5),' corresponding to a value of k=',
```

```
## [1] "The maximum LOO Predictive R-squared is 0.28275 corresponding to a value of k=59."
```

There are a number of reasons k is larger than in the lecture and tutorial examples. First and foremost, the data is relatively large with 32,437 observations. Higher values of k help avoid overfitting concerns, especially when the variance in the data is so large as seen from the graphs above. In addition, since age is so much larger in scale than the other variables, it will dominate the others in distance calculations. Intuitively, many observations have similar values for the explanatory variables in large datasets, and in this case many observations have similar values for the age variable. Using more neighbours to form predictions may allow for more information on the other variables to inform predictions.

Question 4

```
knn_pred = knn.reg(train=df[,c('age','educ','marry')],y=df$logearn,test=hyp_df[,c('age','educ','marry')]
hyp_df$knn_pred = knn_pred
hyp_df
```

```
##    age educ marry      fit      lwr      upr knn_pred
## 1   30    4     0 6.847432 5.343341 8.351524 6.884554
## 2   40    4     0 7.069697 5.565585 8.573809 7.114094
## 3   50    4     0 7.122501 5.618366 8.626637 7.183355
## 4   60    4     0 7.005845 5.501686 8.510003 7.003590
## 5   70    4     0 6.719728 5.215383 8.224073 6.846174
```

```
## 6   30    5     0 7.006263 5.502064 8.510463 7.167068
## 7   40    5     0 7.228528 5.724318 8.732738 7.303477
## 8   50    5     0 7.281332 5.777109 8.785555 7.481590
## 9   60    5     0 7.164676 5.660439 8.668913 7.389177
## 10  70    5     0 6.878559 5.374144 8.382973 6.864096
## 11  30    4     1 7.001807 5.497709 8.505905 6.975493
## 12  40    4     1 7.224072 5.720002 8.728142 7.310041
## 13  50    4     1 7.276876 5.772808 8.780944 7.374003
## 14  60    4     1 7.160220 5.656132 8.664308 7.111821
## 15  70    4     1 6.874103 5.369808 8.378397 7.034444
## 16  30    5     1 7.160638 5.656443 8.664833 7.174824
## 17  40    5     1 7.382903 5.878745 8.887060 7.362183
## 18  50    5     1 7.435707 5.931562 8.939852 7.486799
## 19  60    5     1 7.319050 5.814895 8.823206 7.341017
## 20  70    5     1 7.032933 5.528580 8.537287 7.011373
```

The KNN predictions differ significantly from the OLS predictions for some individuals. In general, education and marital status do not seem to have a monotonic effect on earnings. That education does not is quite surprising given the graph above. This is likely due to the fact that age is dominating distance calculations when we compute nearest neighbours, and does not allow for education and marital status to be influential in predictions.

Question 5

We will not scale marry since it is already a dummy variable.

```
means = colMeans(df[,c('age','educ')])
means
```

```
##       age      educ
## 43.230724  3.019761
```

```
SDs = apply(df[,c('age','educ')],2,sd)
SDs
```

```
##       age      educ
## 13.330749  1.163482
```

```
scaleX = data.frame(scale(df[,c('age','educ')],means,SDs))
scaleX$marry = df$marry
```

Now we can run the KNN model on these scaled variables.

```
r_sq_sc = c()
for (i in ks){
    r2pred = knn.reg(train=scaleX,y=df$logearn,k=i)$R2Pred
    r_sq_sc = c(r_sq_sc,r2pred)
}
optk_sc = ks[which.max(r_sq_sc)]
paste0('The maximum LOO Predictive R-squared is ',round(max(r_sq_sc),5),' corresponding to a value of k=
```

```
## [1] "The maximum LOO Predictive R-squared is 0.28556 corresponding to a value of k=58."
```

While the optimal k does not change much (59 vs. 58), the R-squared is a bit higher (0.283 vs 0.286), so the scaling seems to have helped a little bit. We can also try using "normalization" instead of "standardization" as we have currently done. This way, the variables would all range from 0-1.

```
normalize = function(X){
  scale(X, apply(X, 2, min), apply(X, 2, max) - apply(X, 2, min))
```

```
}
normX = normalize(df[,c('age','educ','marry')])
```

Note that this function does not do anything to the marry dummy variable.

```
r_sq_nm = c()
for (i in ks){
    r2pred = knn.reg(train=normX,y=df$logearn,k=i)$R2Pred
    r_sq_nm = c(r_sq_nm,r2pred)
}
optk_nm = ks[which.max(r_sq_nm)]
paste0('The maximum LOO Predictive R-squared is ',round(max(r_sq_nm),5),' corresponding to a value of k=
```

```
## [1] "The maximum LOO Predictive R-squared is 0.28622 corresponding to a value of k=78."
```

The optimal k seems to have changed a lot, but the R-squared is actually a bit lower than when using scaling via standardization. The status quo approach when scaling variables is almost always standardization, although using KNN when dummy variables are needed is not generally recommended. In either case, scaling variables allows KNN to consider changes in the educ and marry variables as having relatively more importance to predictions than without scaling (due to distance calculations), which explains why the model performs better. As the graphs and OLS results showed, these are important predictors of earnings.

Question 6

```
hyp_scale = data.frame(scale(hyp_df[,c('age','educ')],means,SDs))
hyp_scale$marry = hyp_df$marry
knn_pred_sc = knn.reg(train=scaleX,y=df$logearn,test=hyp_scale,k=optk_sc)$pred
hyp_df$knn_pred_sc = knn_pred_sc
hyp_df
```

```
##    age educ marry      fit      lwr      upr knn_pred knn_pred_sc
## 1   30    4     0 6.847432 5.343341 8.351524 6.884554    6.957259
## 2   40    4     0 7.069697 5.565585 8.573809 7.114094    7.079082
## 3   50    4     0 7.122501 5.618366 8.626637 7.183355    7.170779
## 4   60    4     0 7.005845 5.501686 8.510003 7.003590    7.063837
## 5   70    4     0 6.719728 5.215383 8.224073 6.846174    6.675656
## 6   30    5     0 7.006263 5.502064 8.510463 7.167068    7.156564
## 7   40    5     0 7.228528 5.724318 8.732738 7.303477    7.182209
## 8   50    5     0 7.281332 5.777109 8.785555 7.481590    7.181484
## 9   60    5     0 7.164676 5.660439 8.668913 7.389177    7.388531
## 10  70    5     0 6.878559 5.374144 8.382973 6.864096    6.904642
## 11  30    4     1 7.001807 5.497709 8.505905 6.975493    6.955487
## 12  40    4     1 7.224072 5.720002 8.728142 7.310041    7.222872
## 13  50    4     1 7.276876 5.772808 8.780944 7.374003    7.352891
## 14  60    4     1 7.160220 5.656132 8.664308 7.111821    7.050485
## 15  70    4     1 6.874103 5.369808 8.378397 7.034444    6.941699
## 16  30    5     1 7.160638 5.656443 8.664833 7.174824    7.200710
## 17  40    5     1 7.382903 5.878745 8.887060 7.362183    7.307843
## 18  50    5     1 7.435707 5.931562 8.939852 7.486799    7.610073
## 19  60    5     1 7.319050 5.814895 8.823206 7.341017    7.333140
## 20  70    5     1 7.032933 5.528580 8.537287 7.011373    6.902468
```

The predictions seem to differ quite a bit from the KNN model that does not scale. The changes in predictions moving to higher education levels and marital status seem to make more sense, with both generally having a positive effect on earnings.