2nd_Harvard_project

Konstantinos Roinas

1/6/2021

INTRODUCTION

In this project we will work with adult.csv dataset from kaggle. https://www.kaggle.com/wenruliu/adult-income-dataset (https://www.kaggle.com/wenruliu/adult-income-dataset)

For the needs of our project we have upload this csv in this url
https://github.com/kroinas/2ndproject_edx/raw/main/adult.csv
(https://github.com/kroinas/2ndproject_edx/raw/main/adult.csv)

So lets load our dataset

```
urlfile="https://github.com/kroinas/2ndproject_edx/raw/main/adult.csv"
income<-read.csv(url(urlfile))
```

Lets see the dimensions and class of our dataset.

```
dim(income)
```

```
## [1] 32561    15
```

```
class(income)
```

```
## [1] "data.frame"
```

we see that we have a dataframe of 32561 samples and 15 columns. Lets see our columns.

```
head(income)
```

```
##   age workclass fnlwgt    education education.num marital.status
## 1  90         ?  77053      HS-grad             9        Widowed
## 2  82   Private 132870      HS-grad             9        Widowed
## 3  66         ? 186061 Some-college            10        Widowed
## 4  54   Private 140359      7th-8th             4       Divorced
## 5  41   Private 264663 Some-college            10      Separated
## 6  34   Private 216864      HS-grad             9       Divorced
##          occupation  relationship  race    sex capital.gain capital.loss
## 1                 ? Not-in-family White Female            0         4356
## 2   Exec-managerial Not-in-family White Female            0         4356
## 3                 ?     Unmarried Black Female            0         4356
## 4 Machine-op-inspct     Unmarried White Female            0         3900
## 5    Prof-specialty     Own-child White Female            0         3900
## 6     Other-service     Unmarried White Female            0         3770
##   hours.per.week native.country income
## 1             40  United-States  <=50K
## 2             18  United-States  <=50K
## 3             40  United-States  <=50K
## 4             40  United-States  <=50K
## 5             40  United-States  <=50K
## 6             45  United-States  <=50K
```

We have columns like race, sex,marital status,education, hours.per.week working etc. and finally a column income. Lets see what can be the contents of this column.

```
unique(income$income)
```

```
## [1] "<=50K" ">50K"
```

There are only 2 possible values in income <=50K (low) and >50K (high).

The scope of this project will be to estimate the income using some of the columns as predictors.

ANALYSIS

Lets try produce some graphs now to check distributions of our samples.

For this and for sure throughout the project we will need package tidyverse in case is not installed.

```
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: tidyverse
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.0     v purrr   0.3.4
## v tibble  3.0.1     v dplyr   1.0.0
## v tidyr   1.1.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
```

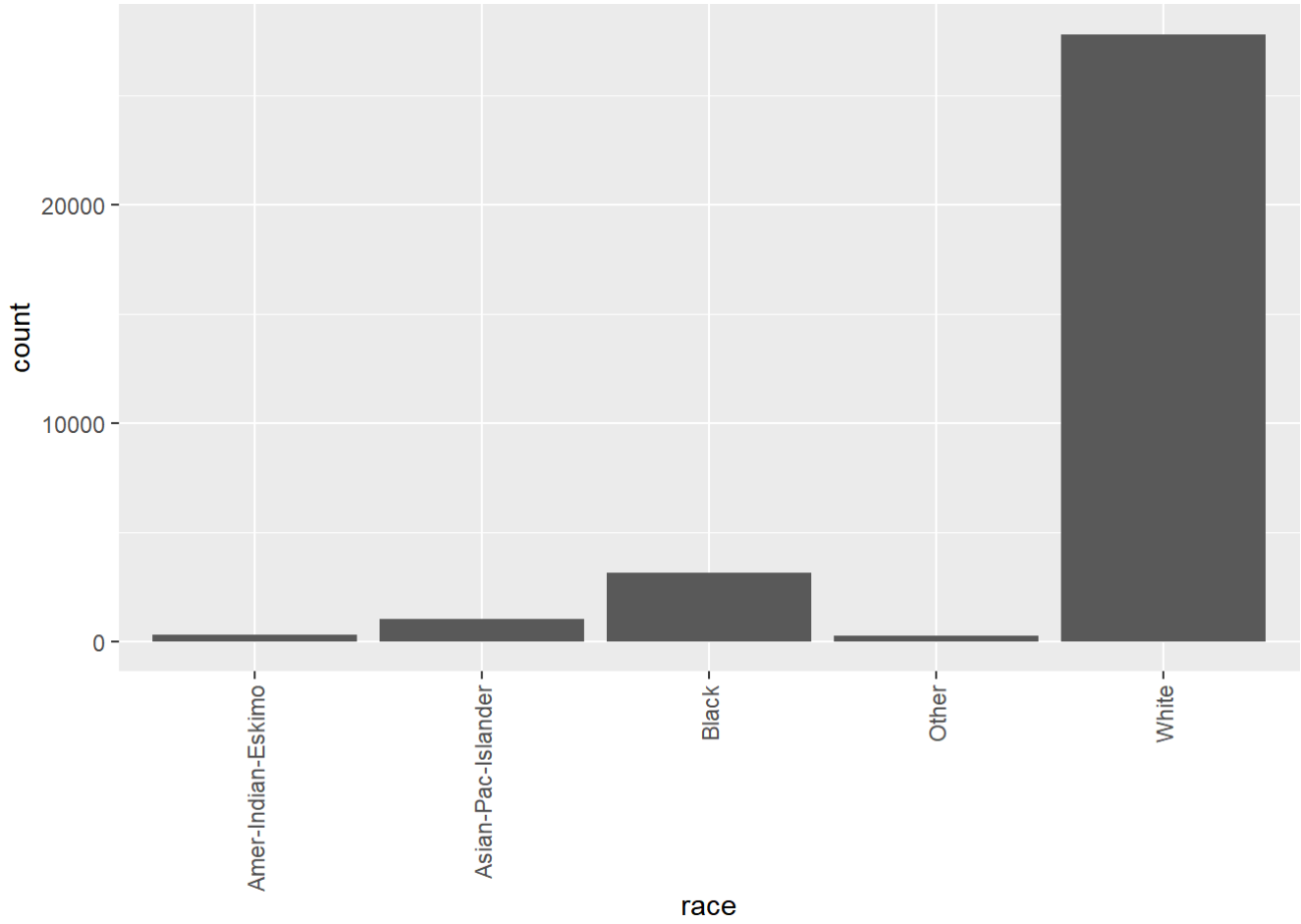
```
## Warning: package 'stringr' was built under R version 4.0.5
```

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

```
library(tidyverse)
```

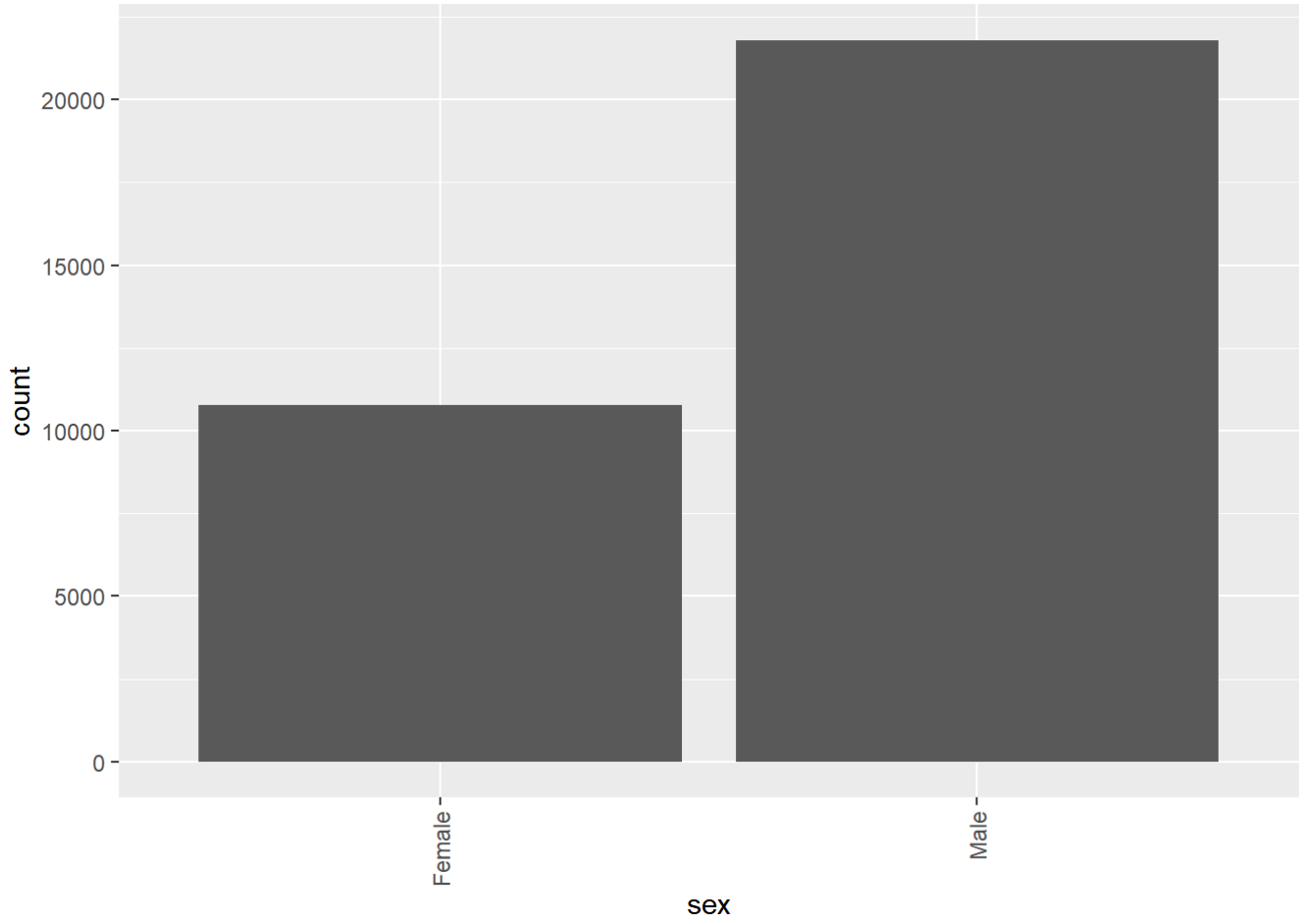
```
income%>%group_by(race)%>%summarize(race,count=n())%>%ggplot(aes(race))+geom_bar()+theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
```

```
## `summarise()` regrouping output by 'race' (override with `.groups` argument)
```



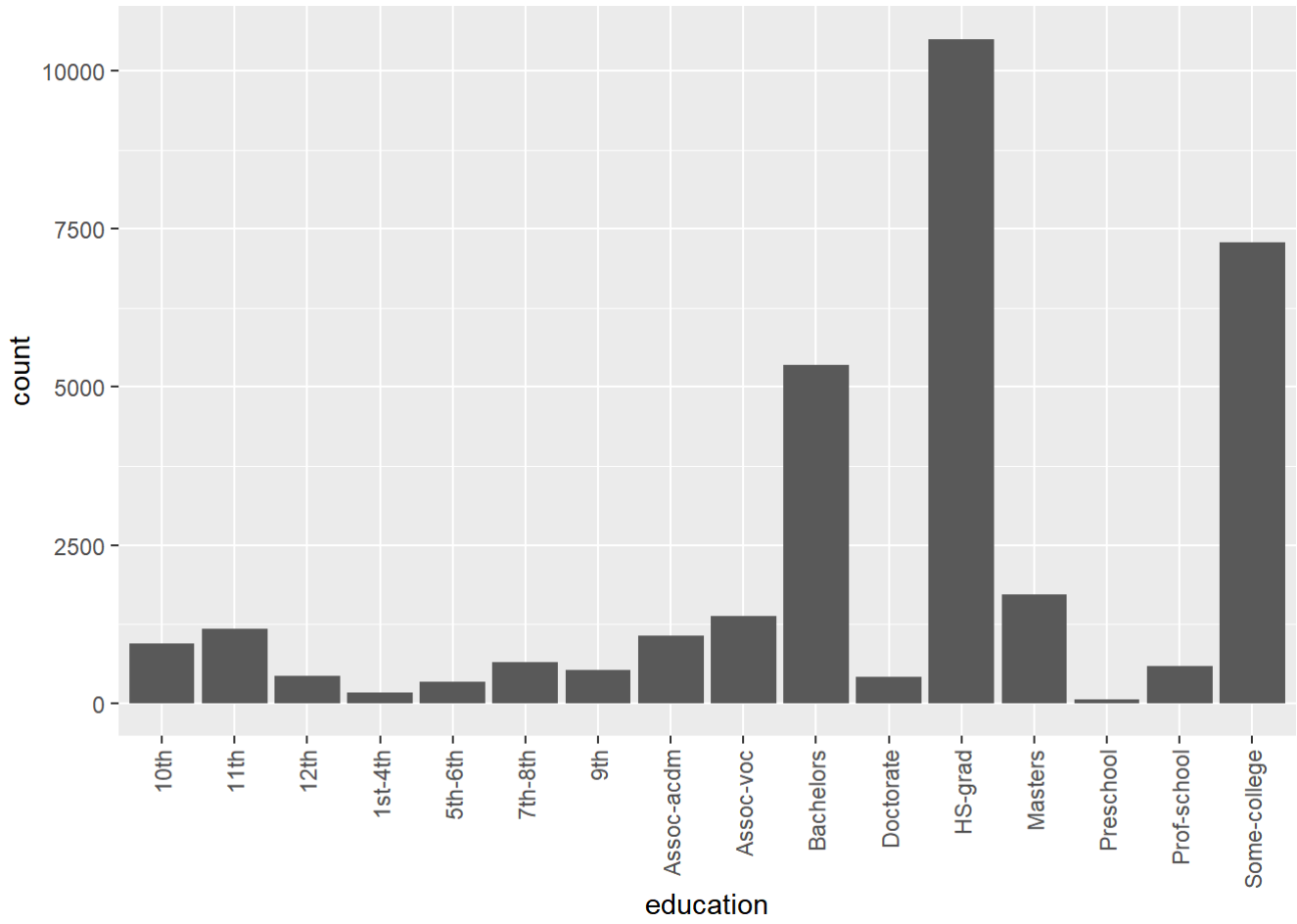

```
income%>%group_by(sex)%>%summarize(sex,count=n())%>%ggplot(aes(sex))+geom_bar()+theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
```

```
## `summarise()` regrouping output by 'sex' (override with `.groups` argument)
```



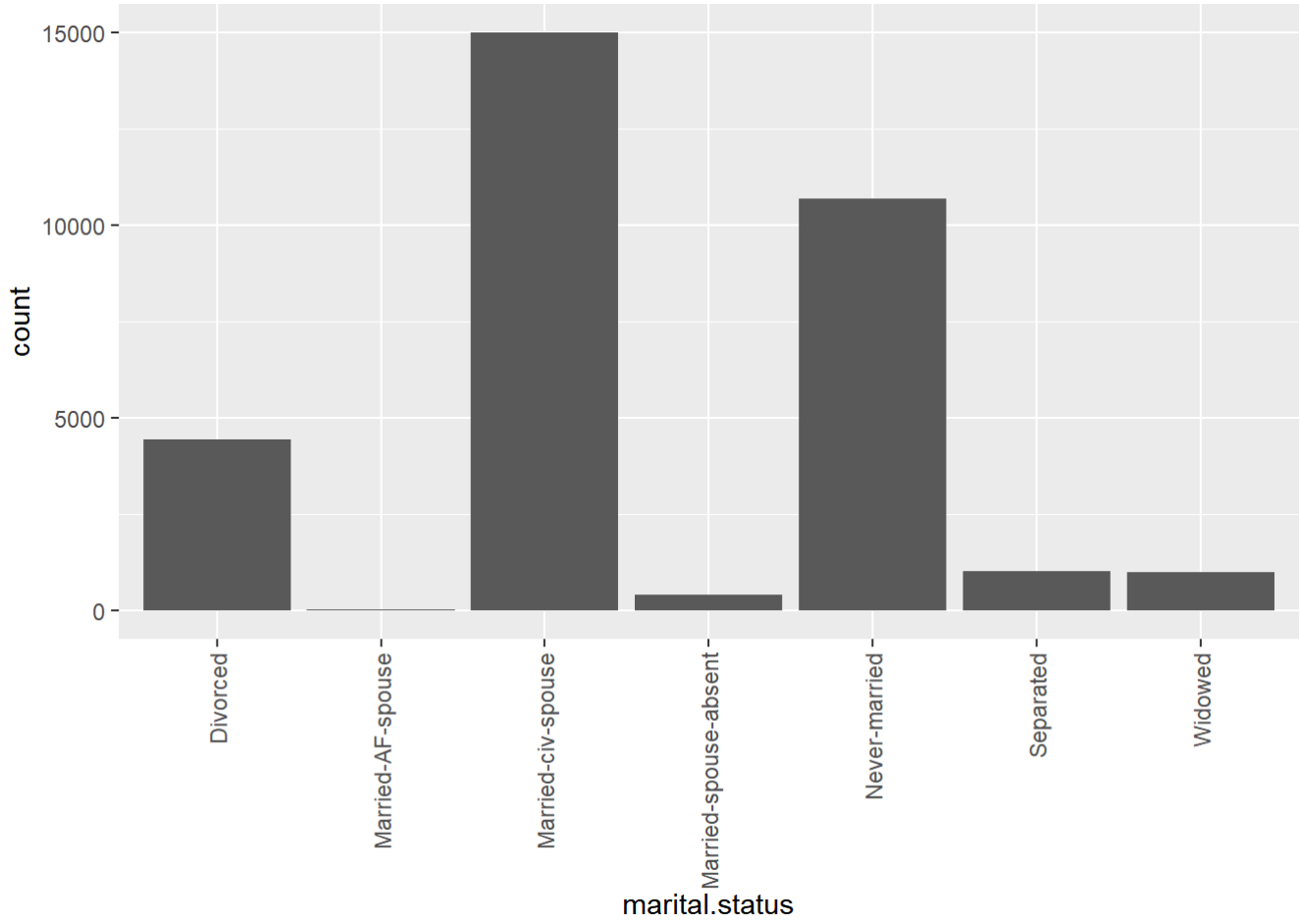

```
 income%>%group_by(education)%>%summarize(education,count=n())%>%ggplot(aes(education))+geom_
bar()+theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
```

```
## `summarise()` regrouping output by 'education' (override with `.groups` argument)
```



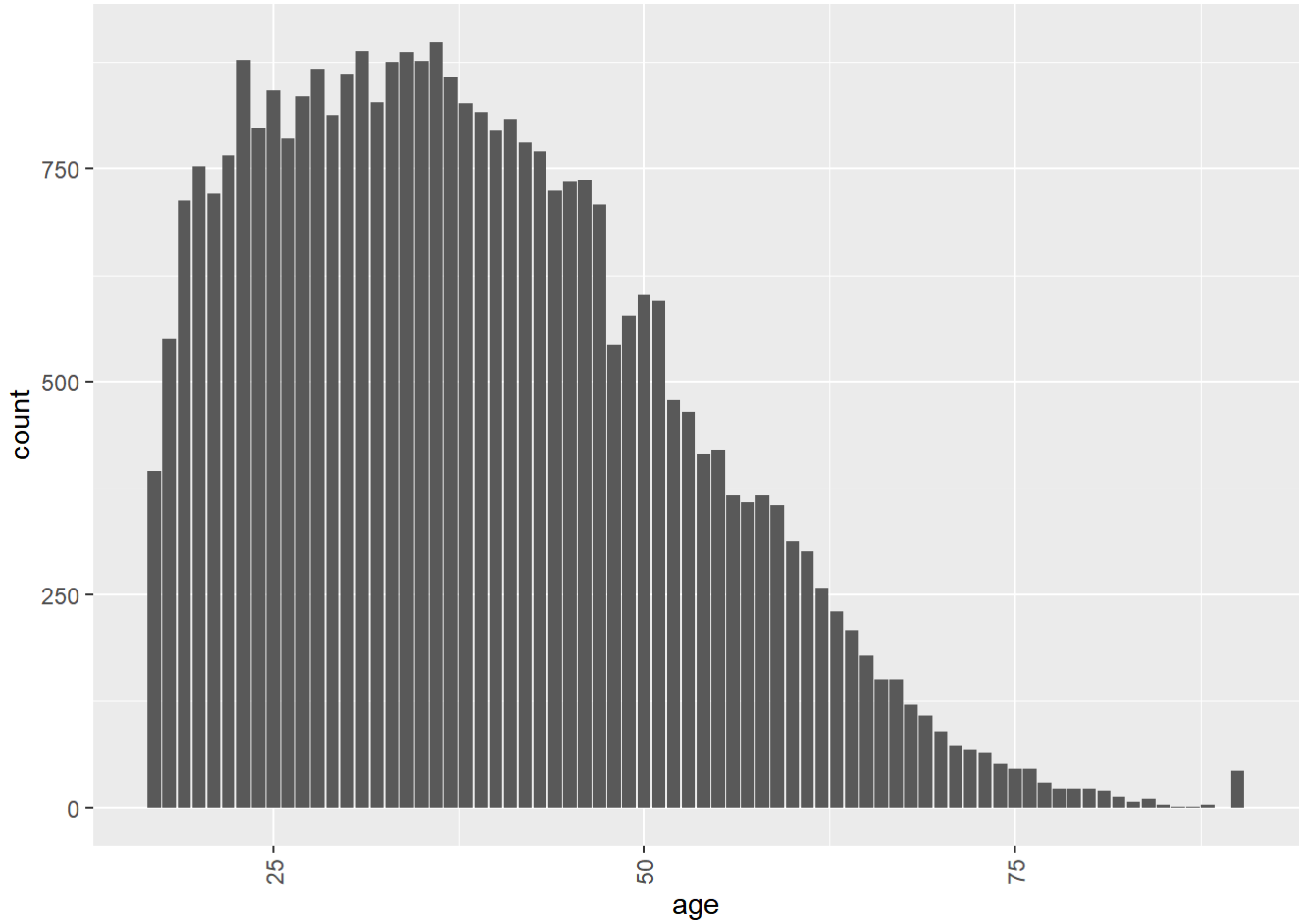

```
 income%>%group_by(marital.status)%>%summarize(marital.status,count=n())%>%ggplot(aes(marita
l.status))+geom_bar()+theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
```

```
## `summarise()` regrouping output by 'marital.status' (override with `.groups` argument)
```




```
 income%>%group_by(age)%>%summarize(age,count=n())%>%ggplot(aes(age,width=10))+geom_bar()+the
me(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
```

```
## `summarise()` regrouping output by 'age' (override with `.groups` argument)
```



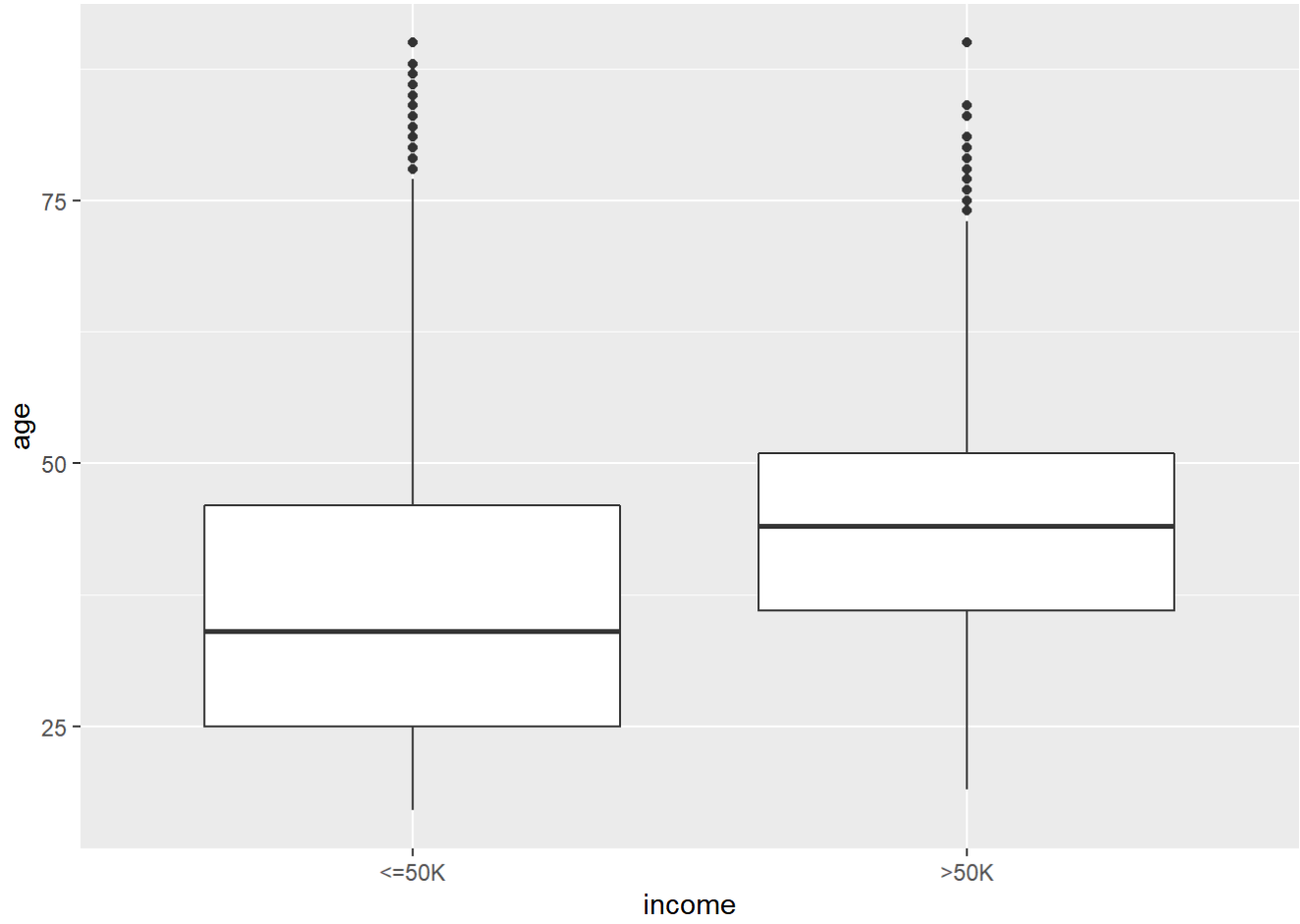
We see that two thirds of our samples are men and the majority also belong to white race. With angle 90 we turn x text 90 degrees in order not overlap.

FINDING OUR PREDICTORS

Lets see if age affects income

```
income%>%group_by(age)%>%summarize(age,income)%>%ggplot(aes(income,age))+geom_boxplot()
```

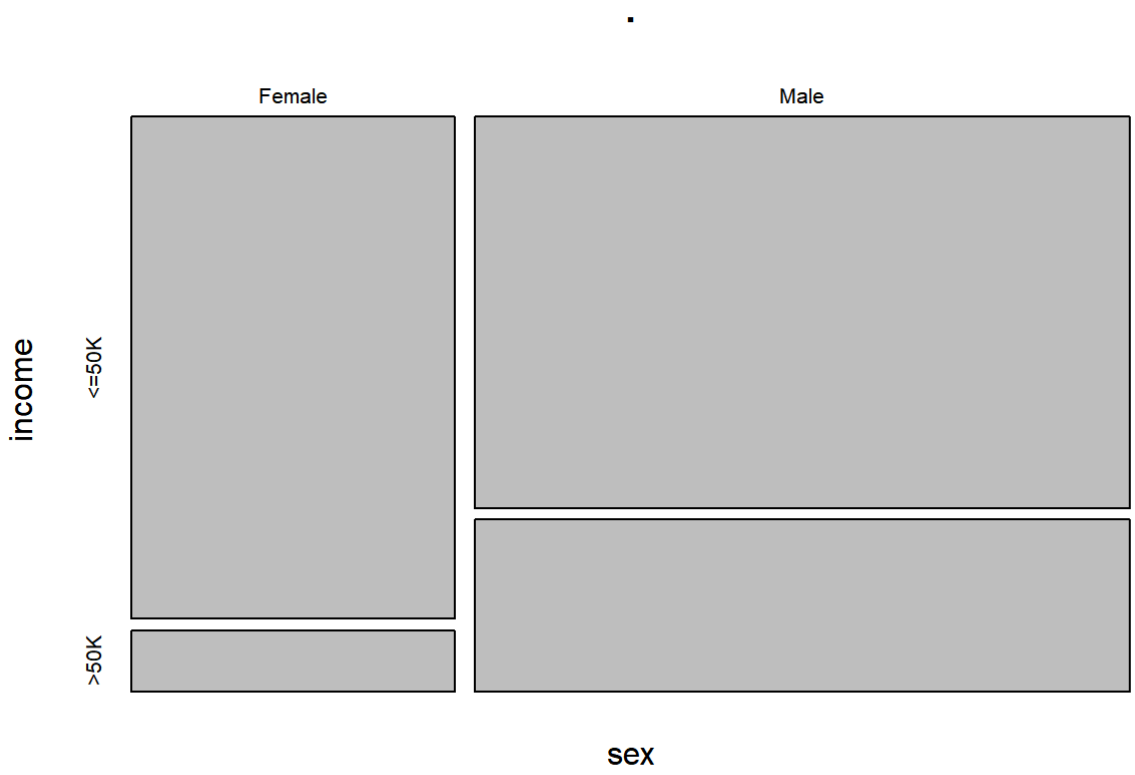
```
## `summarise()` regrouping output by 'age' (override with `.groups` argument)
```



It is clear that income >50K is more usual to older people. So yes age is a predictor.

Lets examine sex now.We will create a 2x2 table setting the dimension names.

```
income_sex<- table(income$sex,income$income)
names(dimnames(income_sex)) <- c("sex", "income")
income_sex%>%plot(sex,income)
```



From this plot is clear high income is more common between men. So that is our second predictor.

Regarding native country we observed that 90% of dataset is ‘United States’.

```
dim(income%>%filter(native.country=="United-States"))
```

```
## [1] 29170    15
```

So we do not think worth to take it into account as predictor.

We will examine now if race affects income

```
income_race<- table(income$race,income$income)
names(dimnames(income_race)) <- c("race", "income")
```

Trying to find the percentage of low income (<=50K) in each race.

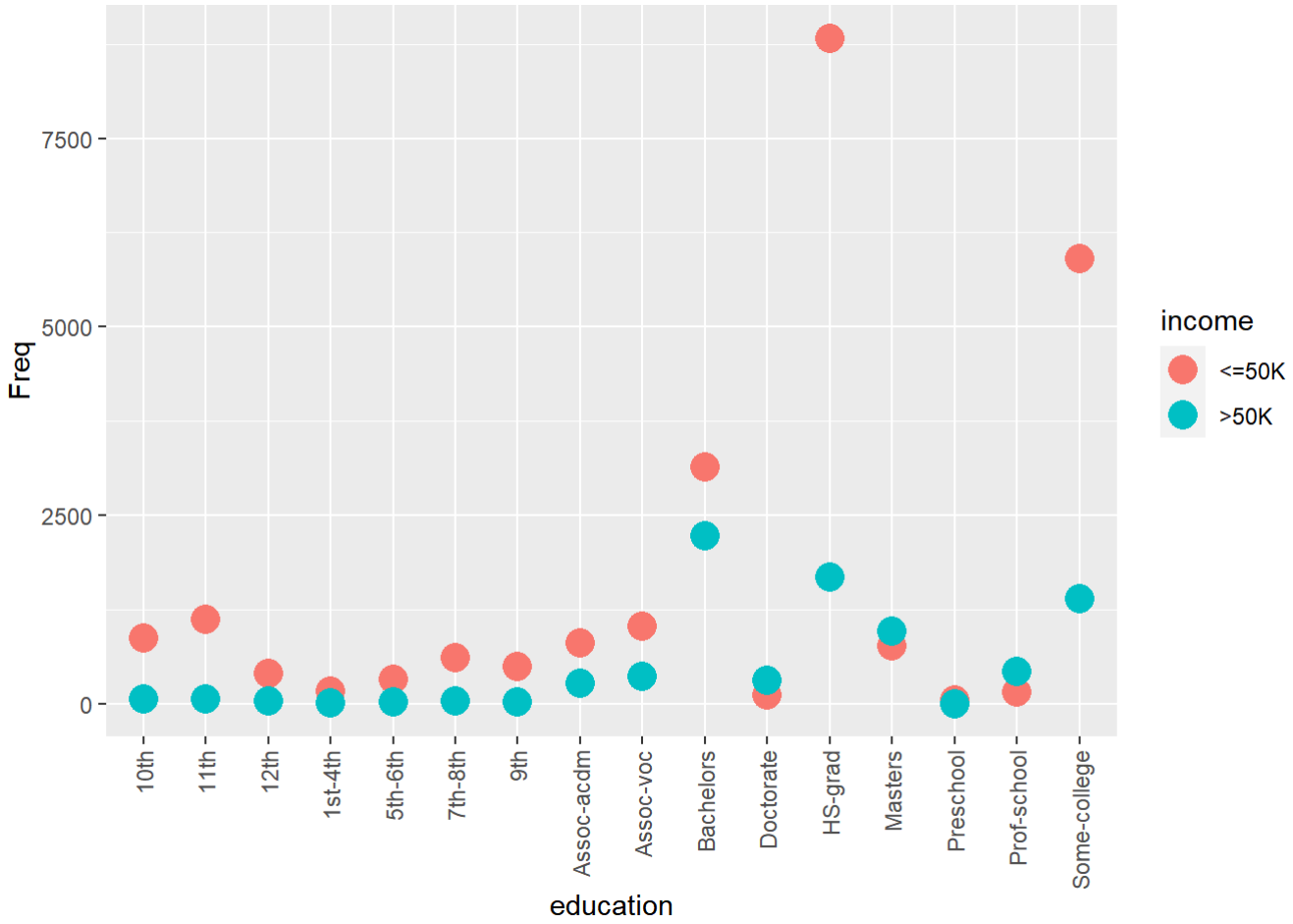
```
income_race<-cbind(income_race,rowSums(income_race))
percentage_low_income<-income_race[,1]/income_race[,3]*100
as.data.frame(percentage_low_income)
```

```
##                     percentage_low_income
## Amer-Indian-Eskimo              88.42444
## Asian-Pac-Islander              73.43600
## Black                           87.61204
## Other                           90.77491
## White                           74.41401
```

Here we divided first column (low income number of samples per race) with the third one that is the total number of samples per race. And transform it to dataframe for a better visualization. We see for example 74 of 'white' or 'Asian-Pac-Islander' with low income while this reach 87-90 on rest races. So yes race is a predictor.

Another column that we suspect as predictor is education. Lets plot high and low incomes appearances in our dataset depending on education.

```
income_education<- table(income$education,income$income)
income_education<-as.data.frame(income_education)
colnames(income_education)<-c("education","income","Freq")
income_education%>%group_by(education)%>%ggplot(aes(education,Freq,fill=income))+geom_point(a
es(color=income), size=5)+ theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1
))
```



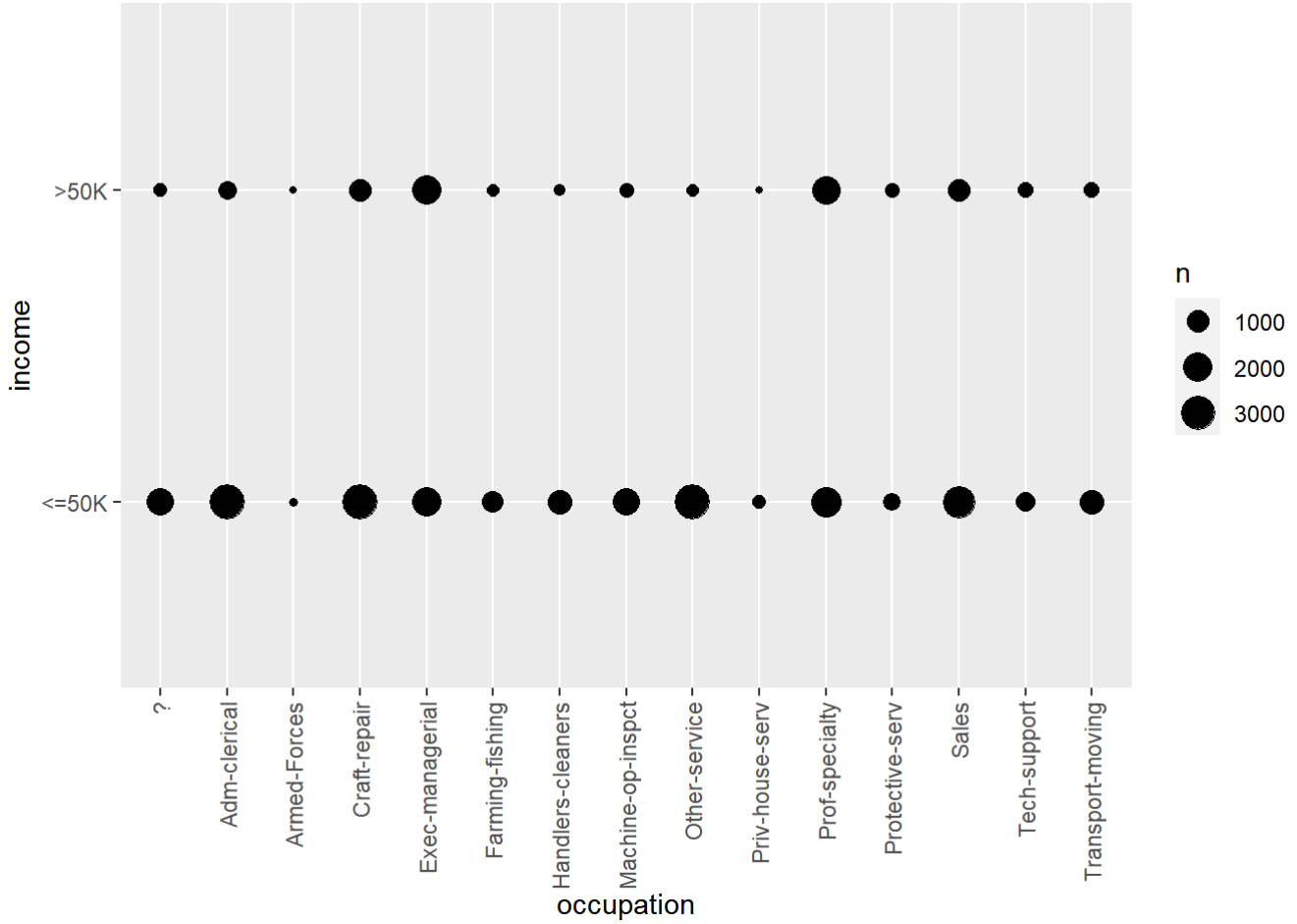
We distinguish color with income and enlarge our points for having clear the income with size parameter.

We see that for some education classes (like Doctorate, masters) high income is more common, while in the majority no. Also in some of the rest the two incomes are close (like 1st-4th or 5th-6th), while in ohers low incomes are much more common (like in HS-grad or Some-collge). So yes education stands for another predictor.

A final variable we suspect is occupation. Lets plot a figure to realise if could be predictor.

```
income%>%group_by(occupation)%>%summarize(occupation,income)%>%ggplot(aes(occupation,income))
+geom_count()+theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
```

```
## `summarise()` regrouping output by 'occupation' (override with `.groups` argument)
```



Here n stands for the number of lines in our dataset. It is clear that while for some occupation classes (like Exec-managerial or Prof-specialty) numbers between the 2 classes of income are similar, in other occupations (Adm-clerical for example) the difference is huge (in favor of low income). So yes seems to work as a predictor also.

APPLYING ML MODELS / RESULTS

It is now time build our models using the predictors announced before. We will use 2 models and these ones will be knn and rpart (decision tree).

First of all we have to load caret library if it is not existing as well as create our train and test data sets. For the last according to Pareto principle is better use a ratio 80/20 (80 train).

```
if(!require(caret)) install.packages("caret",repos="http://cran.us.r-project.org")
```

```
## Loading required package: caret
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

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

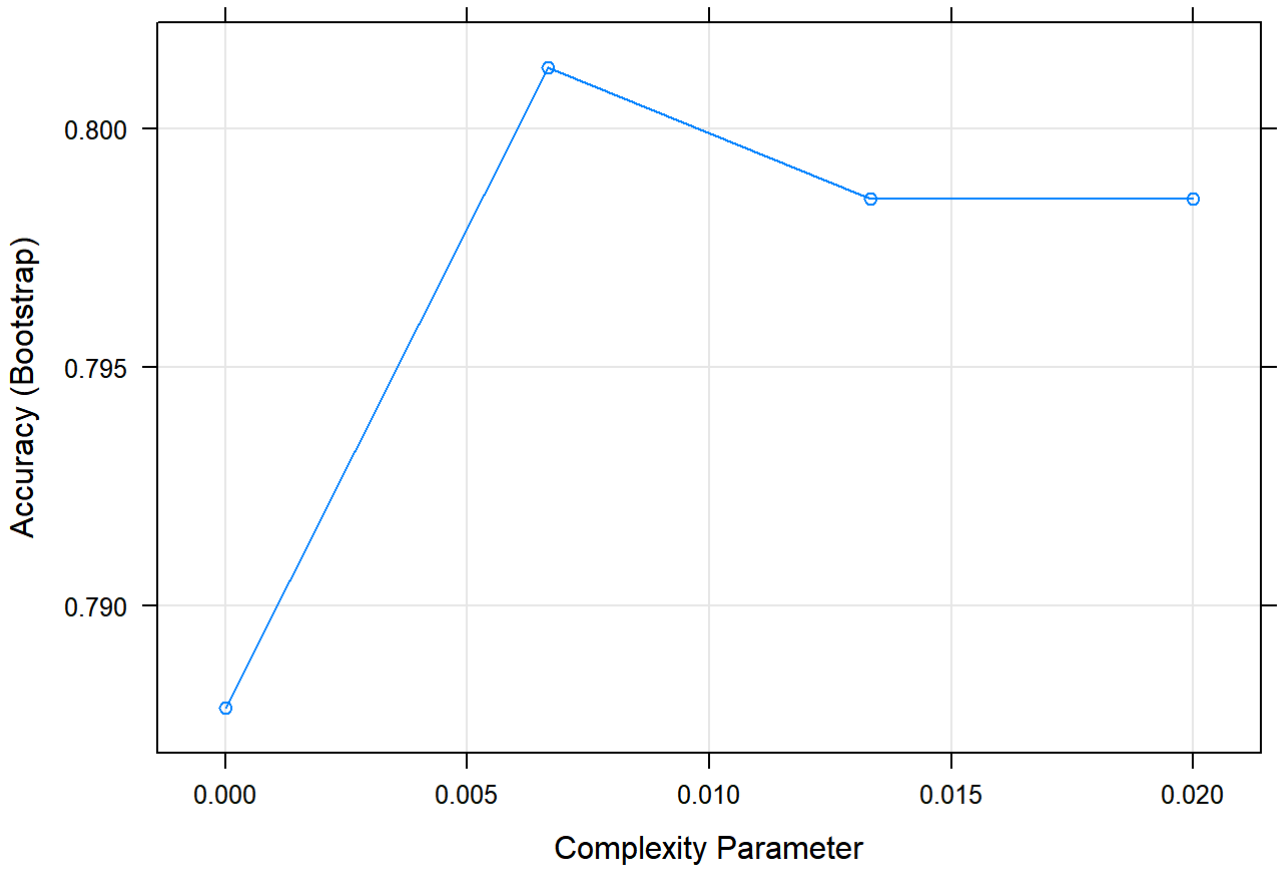
```
library(caret)
test_index <- createDataPartition(income$income, times = 1, p = 0.2, list=FALSE)
train_set <- income%>% slice(-test_index)
test_set<-income%>% slice(test_index)
```

RPART TRAINING

```
train_rpart <- train(income~ education+age+race+sex+occupation,method = "rpart", tuneGrid = d
ata.frame(cp = seq(0, 0.02, len = 4)),data = train_set)
```

cp is a parameter used in decision trees that more or less tells us till where will continue splitting. It is used to control the size of the decision tree and to select the optimal tree size. If the cost of adding another variable to the decision tree from the current node is above the value of cp, then tree building does not continue.

```
plot(train_rpart)
```



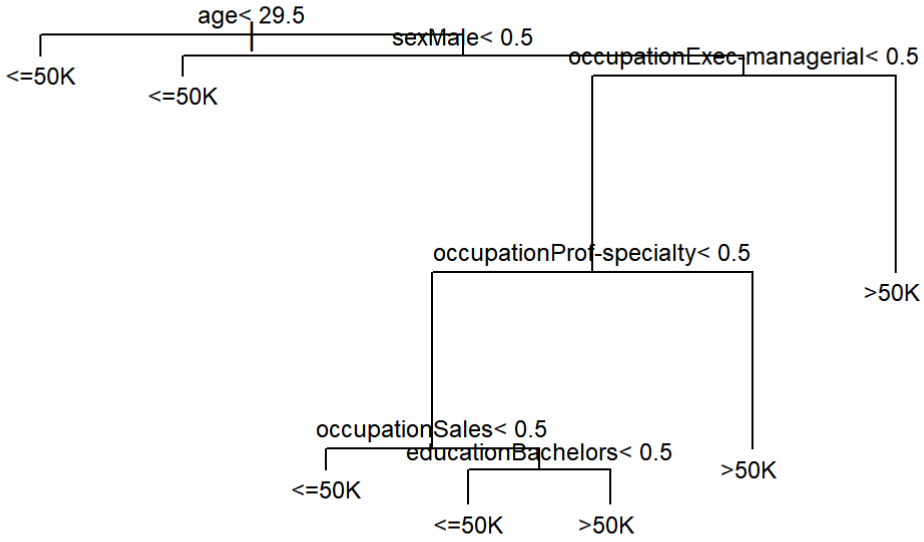

```
train_rpart$bestTune
```

```
##            cp
## 2 0.006666667
```

We see the optimal value of cp is 0.006666667 and gives an accuracy just above 80%.

To visualize our tree we give the following commands:

```
plot(train_rpart$finalModel,margin = 0.1)
text(train_rpart$finalModel,cex = 0.75)
```



To explain a bit how tree works, it is asked if age<29.5. If answer is yes we go left, so income is <=50K. If answer is no we go right finding another question. SexMale<0.5? This means is Female? If answer is yes again go left and find income <=50K. If not go right and find the question if is not(<0.5..) Exec-managerial in occupation and so on.

KNN TRAINING

In k-NN classification, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors

```
train_knn <- train(income ~ education+age+race+sex+occupation,method = "knn", tuneGrid = dat
a.frame(k = seq(15, 51, 2)),data = train_set)
train_knn$bestTune
```

```
##     k
## 12 37
```

```
y_hat_knn<-predict(train_knn,test_set)
confusionMatrix(y_hat_knn,as.factor(test_set$income))$overall["Accuracy"]
```

```
##  Accuracy
## 0.8027023
```

Here we see the best k is from the last values. In general we set odd values not too big not too small.

Then we used our trained model to predict the income for test set and finally we calculated accuracy.

We used as.factor inside matrix for test_set$income, as it is required to be of this class the values we compare (while its normal class is ‘character’).

```
class(test_set$income)
```

```
## [1] "character"
```

This model also gives us 80% accuracy as we see (similar to the one of rpart).

TRYING INCREASE MODELS’ ACCURACY

In order identify variables that affect income, we run a simple rf (random forest) to take advantage of its varImp feature. This feature indicates which variable affects more our result.

```
train_rf <- train(income ~ ., method = "rf", data=income)
varImp(train_rf)
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 100)
##
##                                 Overall
## marital.statusMarried-civ-spouse 100.000
## fnlwgt                            97.870
## age                               73.491
## capital.gain                      73.183
## education.num                     62.974
## hours.per.week                    41.765
## capital.loss                      22.422
## marital.statusNever-married       10.835
## occupationExec-managerial          8.855
## workclassPrivate                   7.028
## occupationProf-specialty           6.926
## sexMale                            5.689
## workclassSelf-emp-not-inc          5.648
## occupationSales                    5.062
## occupationCraft-repair             5.058
## workclassLocal-gov                 3.904
## raceWhite                          3.867
## native.countryUnited-States        3.782
## relationshipWife                   3.710
## workclassSelf-emp-inc              3.694
```

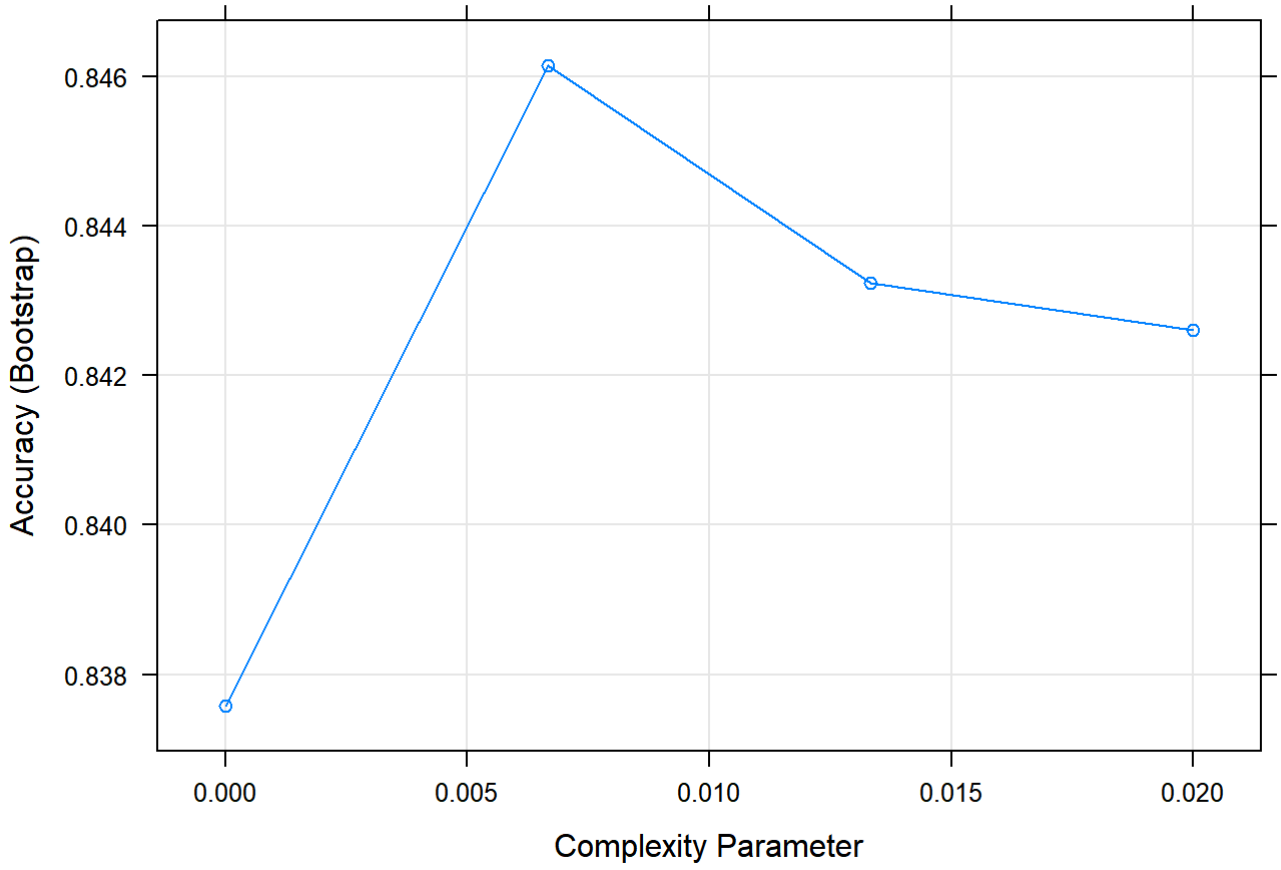
We see that income is affected strongly from marital.status, capital.gain or loss. Education.num is related to education that already use. Age is within the results of varImp, but race or sex are very low. Capital gain or loss could be substituted from capital.profit=capital.gain-capital.loss So from our models we will throw away sex and race and will add capital.profit,marital.status,hours.per.week (the more you work the more the income sounds logical). Here we have to mention fnlwgt. We really do not understand why is here as it is the finalweight. The number of real population that represents each line. We will ignore it.

But lets create first the profit in both train and test sets

```
train_set<-train_set%>%mutate(capital.profit=capital.gain-capital.loss)
test_set<-test_set%>%mutate(capital.profit=capital.gain-capital.loss)
```

It is time train our first model (rpart) with the predictor changes

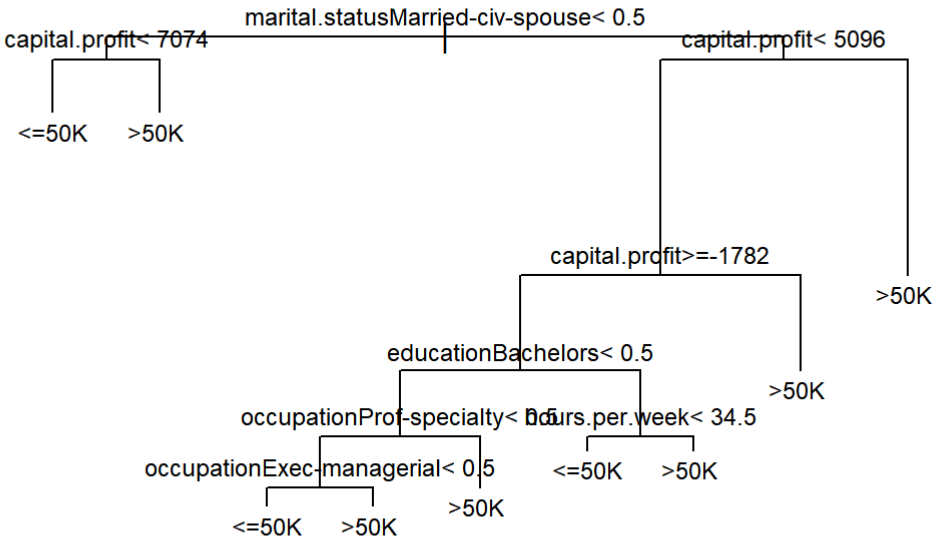
```
train_rpart <- train(income~ education+age+occupation+capital.profit+marital.status+hours.per.week,method = "rpart", tuneGrid = data.frame(cp = seq(0, 0.02, len = 4)),data = train_set)
plot(train_rpart)
```




```
y_hat_rpart<-predict(train_rpart,test_set)
confusionMatrix(y_hat_rpart,as.factor(test_set$income))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction <=50K >50K
##      <=50K  4587  619
##      >50K    357  950
##
##                Accuracy : 0.8501
##                  95% CI : (0.8412, 0.8587)
##     No Information Rate : 0.7591
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5655
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9278
##             Specificity : 0.6055
##          Pos Pred Value : 0.8811
##          Neg Pred Value : 0.7269
##              Prevalence : 0.7591
##          Detection Rate : 0.7043
##    Detection Prevalence : 0.7993
##       Balanced Accuracy : 0.7666
##
##        'Positive' Class : <=50K
##
```

```
plot(train_rpart$finalModel,margin = 0.1)
text(train_rpart$finalModel,cex = 0.75)
```



Trying explain the tree: is marital status Married-civ.spouse? If yes, is capital.profit less than 7074, then income is <=50K and so on.

We passed 84% much better than before (cp same as before).

We will repeat now our second model knn with the updated predictors. We will change a bit k to be always odd but include higher values. We remember last time was the max value 50 more or less.

```
train_knn <- train(income~ education+age+occupation+marital.status+capital.profit+hours.per.w
eek,method = "knn",tuneGrid = data.frame(k = seq(5, 101, 8)), data=train_set)
train_knn$bestTune
```

```
##    k
## 3 21
```

We see this time best k is 21. In general while in rpart cp is very stable, k in knn we have seen it is varying. Also while there is a rule saying best k is sqrt(N) and here N is close to 30000, we did not see applies here (as it had to be k= 170 in our case).

Lets calculate the prediction in test set and check the accuracy.

```
y_hat_knn<-predict(train_knn,test_set)
confusionMatrix(y_hat_knn,as.factor(test_set$income))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction <=50K >50K
##      <=50K  4700  737
##      >50K    244  832
##
##                Accuracy : 0.8494
##                  95% CI : (0.8405, 0.858)
##     No Information Rate : 0.7591
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5387
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9506
##             Specificity : 0.5303
##          Pos Pred Value : 0.8644
##          Neg Pred Value : 0.7732
##              Prevalence : 0.7591
##          Detection Rate : 0.7216
##    Detection Prevalence : 0.8348
##       Balanced Accuracy : 0.7405
##
##        'Positive' Class : <=50K
##
```

CONCLUSIONS

Seems rpart is much faster (training takes 5 mins while knn runs for 2-3 hours) while both are more or less same accurate close to 85%.

Here to mention that while was not rested but as was used to identify the predictors, RF (random forest) was desperately slow in training, taking whole day more or less to run. The HW used was i5 processor with 8 GB ram (DDR3).

We did not set seed, as we wanted run them 2-3 times to see the class of result and not accuracy of second digit for example.

Both seem to have 95% more or less sensitivity and close to 53% specificity. Having as positive scenario income <=50K, sensitivity means the capability of model to correctly identify these cases of low income, while specificity is the capability of model identify the high incomes that are the more rare cases. This happens as the high income samples in dataset are much less.

We have to mention that train control did not provide any improvement on results. We tried a ‘cv’ (cross validation) of 10 folds.This means our training models 10 times split train set (internally in test and train set), differently each time to do a better training. It is a method used to avoid overfitting. Overfitting we have when accuracy is very high more than 90% in train set that falls dramatically in test set. But in our models trainControl did not improve results, we understand we have not overfitting.In general results between 80 and 90% in prediction are considered very good results. We do not believe can do something better on selecting predictors. Never can have a model with 100% accuracy.

What could be done more (as future work continuation of the present one), would be try further ML models like RF,LDA,QDA for example.

We tried provide the maximum detail possible in our steps but on the other hand is not a tutorial of ML but a project to apply models in a given dataset to check their accuracy. Our target was having a right equilibrium on that matter. Hope we managed it.

We have to mention something about the structure of the report. We tried follow the closest possible what was asked but we had a structure in our mind for how to present our work. So maybe some parts that appear like missing (for example key steps) that cannot be found in introduction but of course the reader can find them in analysis section or even in ‘APPLYING ML MODELS’ section.

Finally maybe the use of first plural person used throughout the project sounds strange to some readers but is what usually used by the author. Of course is a personal and not group work.