Class 05 | Regression Trees

BQOM 2578 | Data Mining

Theresa Wohlever

Sunday, September 28, 2025

Table of contents

Loading packages

Lets start by calling some libraries that are useful for building and visualizing trees:

- rpart
- rpart.plot

```
#After installing comment the install.packages commands
#install.packages("rpart")
#install.packages("rpart.plot")
#install.packages("tidyverse")
#install.packages("patchwork")
#install.packages("corrplot")
#
# Load them
library(rpart)
library(rpart.plot)
#And our usual packages
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.2     v tibble    3.3.0
v lubridate 1.9.4     v tidyr     1.3.1
v purrr     1.1.0
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
library(patchwork)
library(corrplot)
```

```
corrplot 0.95 loaded
```

```r
rm(list = ls())
setwd("/Users/theresawohlever/git_repos/BQOM-2578_DataMining/BQOM-2578_DataMining_twohlever/assignments/05")
```

Importing data

We are using cps09mar dataset, you can find the description on the link.

```r
#read.csv will read the csv into a dataframe df, which we can manipulate in R.
df = read.csv("cps09mar.csv", stringsAsFactors = TRUE)
str(df)
```

```
'data.frame':	50742 obs. of  12 variables:
 $ age      : int  52 38 38 41 42 66 51 49 33 52 ...
 $ female   : int  0 0 0 1 0 1 0 1 0 1 ...
 $ hisp     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ education: int  12 18 14 13 13 13 16 16 16 14 ...
 $ earnings : int  146000 50000 32000 47000 161525 33000 37000 37000 80000 32000 ...
 $ hours    : int  45 45 40 40 50 40 44 44 40 40 ...
 $ week     : int  52 52 51 52 52 52 52 52 52 52 ...
 $ union    : int  0 0 0 0 1 0 0 0 0 0 ...
 $ uncov    : int  0 0 0 0 0 0 0 0 0 0 ...
 $ region   : int  1 1 1 1 1 1 1 1 1 1 ...
 $ race     : int  1 1 1 1 1 1 1 1 1 1 ...
 $ marital  : int  1 1 1 1 1 5 1 1 1 1 ...
```

```r
summary(df)
```

```
      age            female            hisp          education    
 Min.   :15.00   Min.   :0.0000   Min.   :0.0000   Min.   : 0.00  
 1st Qu.:33.00   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:12.00  
 Median :42.00   Median :0.0000   Median :0.0000   Median :13.00  
 Mean   :42.13   Mean   :0.4257   Mean   :0.1488   Mean   :13.92  
 3rd Qu.:51.00   3rd Qu.:1.0000   3rd Qu.:0.0000   3rd Qu.:16.00  
 Max.   :85.00   Max.   :1.0000   Max.   :1.0000   Max.   :20.00  
    earnings          hours            week           union        
 Min.   :     1   Min.   :36.00   Min.   :48.00   Min.   :0.00000  
 1st Qu.: 28000   1st Qu.:40.00   1st Qu.:52.00   1st Qu.:0.00000  
 Median : 42000   Median :40.00   Median :52.00   Median :0.00000  
 Mean   : 55092   Mean   :43.83   Mean   :51.88   Mean   :0.02152  
 3rd Qu.: 65000   3rd Qu.:45.00   3rd Qu.:52.00   3rd Qu.:0.00000  
 Max.   :561087   Max.   :99.00   Max.   :52.00   Max.   :1.00000  
     uncov              region           race           marital     
 Min.   :0.000000   Min.   :1.000   Min.   : 1.000   Min.   :1.000  
 1st Qu.:0.000000   1st Qu.:2.000   1st Qu.: 1.000   1st Qu.:1.000  
 Median :0.000000   Median :3.000   Median : 1.000   Median :1.000  
 Mean   :0.002207   Mean   :2.636   Mean   : 1.434   Mean   :2.763  
 3rd Qu.:0.000000   3rd Qu.:4.000   3rd Qu.: 1.000   3rd Qu.:5.000  
 Max.   :1.000000   Max.   :4.000   Max.   :21.000   Max.   :7.000  
```

Note how all variables are integers.

cps09mar is a 2009 Current Population Survey (CPS), holding data of 50742 US household about several labor force characteristics, restricted to those who worked at least 36 hours per week for at least 48 weeks the past year; excluding military.

Our key dependent variable is earnings (total annual wage and salary earnings in dollars). Most variables are self-explanatory, can look at the documentation for more details.

We would prefer to have our dependent variable first, so lets relocate it.

```r
#Move earnings to the front:
df<-df%>%relocate(earnings)
head(df)
```

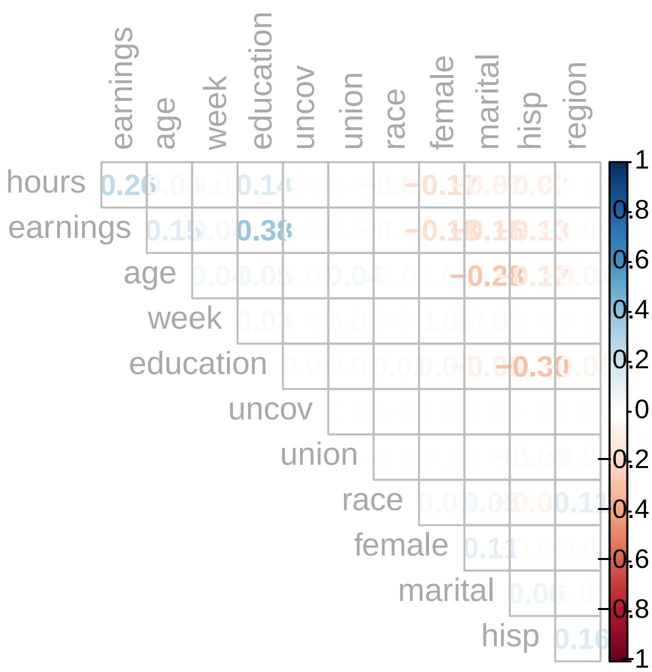
```
  earnings age female hisp education hours week union uncov region race marital
1   146000  52      0    0        12    45   52     0     0      1    1       1
2    50000  38      0    0        18    45   52     0     0      1    1       1
3    32000  38      0    0        14    40   51     0     0      1    1       1
```

4	47000	41	1	0	13	40	52	0	0	1	1	1
5	161525	42	0	0	13	50	52	1	0	1	1	1
6	33000	66	1	0	13	40	52	0	0	1	1	5

Preliminary Analysis

Let's begin as before with evaluating the Correlation Matrix (see questions after code block).

```
#Make and display a correlation matrix:
cormat <- round(cor(df),2)



corrplot(cormat, method="number", type="upper",
  order="AOE",
  tl.col="darkgrey",
  cl.align.text = "r",
  diag=FALSE,
  number.cex=0.9)
```



What are the most influential variables with regard to earnings?

Age, Female, Hispanic, Education, Hours, Marital.

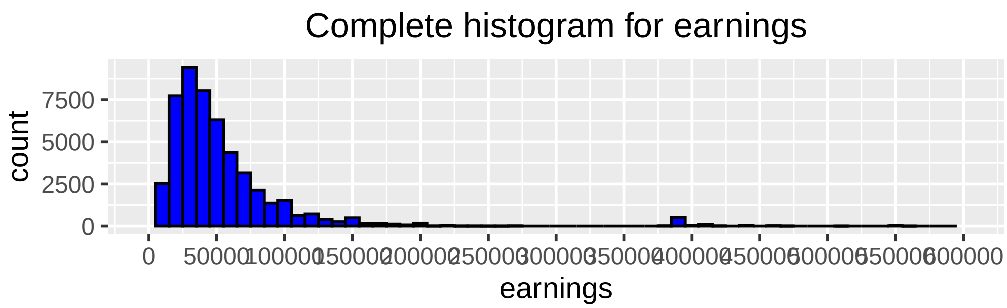
But, how do we interpret the Martial correlation?

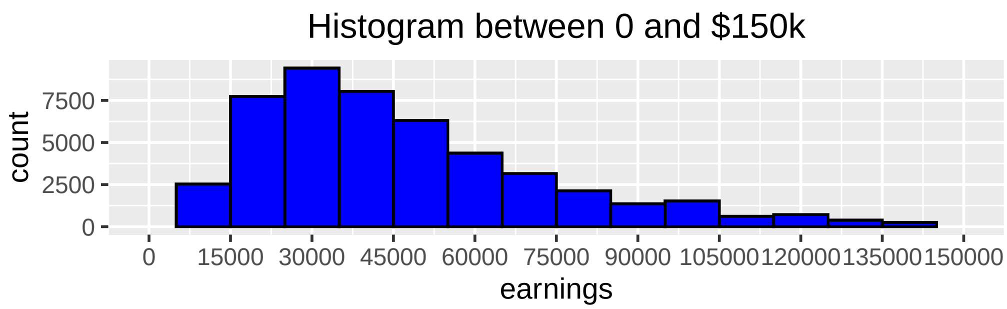
Please note:

- correlation is less of a worry in Regression Trees. The method "takes care of it"; if one variable is picked at some point to make a split and the other (highly correlated) variable does not add any useful additional information, then it will just not be used for a split later.
- We prefer to take a log of the dependent variable (and work with it instead) if the original dependent variable is too skewed. Therefore, you should remember to check the histogram of the dependent variable:

```
#Make a new histogram zooming in to the most frequent values:

g1<-ggplot(df)+aes(x=earnings)+geom_histogram(fill="blue", colour="black", stat="bin", binwidth=10000)+scale_x_continuous(breaks=seq(0

g2<-ggplot(df)+aes(x=earnings)+geom_histogram(fill="blue", colour="black", stat="bin", binwidth=10000)+scale_x_continuous(breaks=seq(0

g1/g2
```

```
Warning: Removed 2 rows containing missing values or values outside the scale range
(`geom_bar()`).

Warning: Removed 1454 rows containing non-finite outside the scale range
(`stat_bin()`).

Warning: Removed 2 rows containing missing values or values outside the scale range
(`geom_bar()`).
```





As suspected, the distribution is not very normal. We can get better predictions by taking log of earnings.

```
df$earnings[1]
```

```
[1] 146000
```

```
log(df$earnings[1])
```

```
[1] 11.89136
```

```
#Get logearnings for the entire dataset

df$logearnings<-log(df$earnings)
summary(df$logearnings)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.00   10.24   10.65   10.66   11.08   13.24
```

```
#For reference:
log(25000)
```

```
[1] 10.12663
```

```r
log(50000)
```

```
[1] 10.81978
```

```r
log(100000)
```

```
[1] 11.51293
```

```r
log(150000)
```

```
[1] 11.91839
```

```r
log(200000)
```

```
[1] 12.20607
```

```r
log(250000)
```

```
[1] 12.42922
```

```r
log(400000)
```

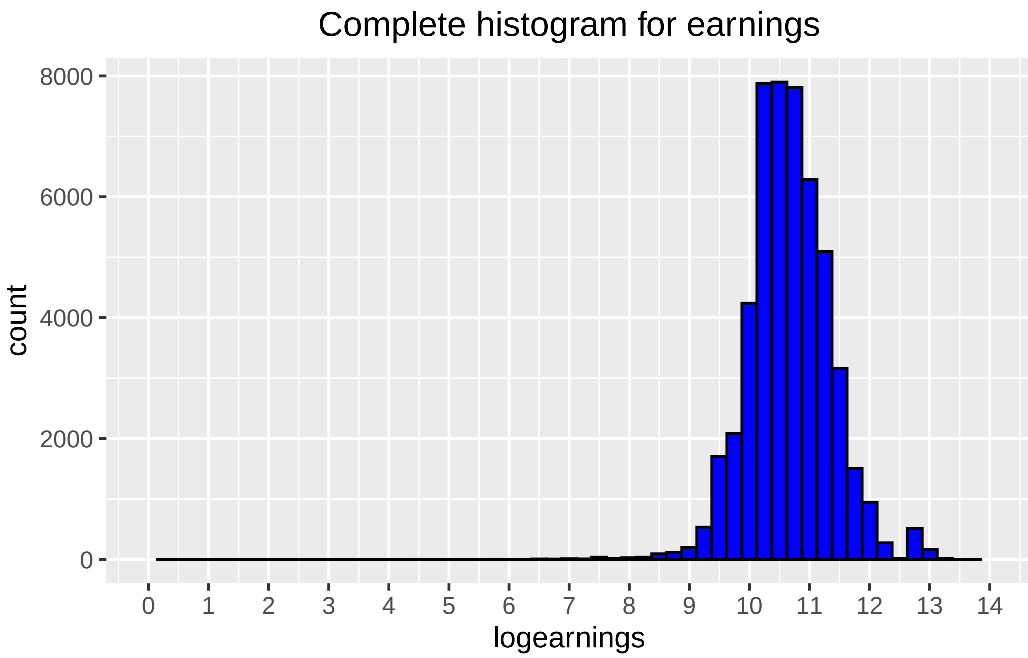
```
[1] 12.89922
```

```r
log(500000)
```

```
[1] 13.12236
```

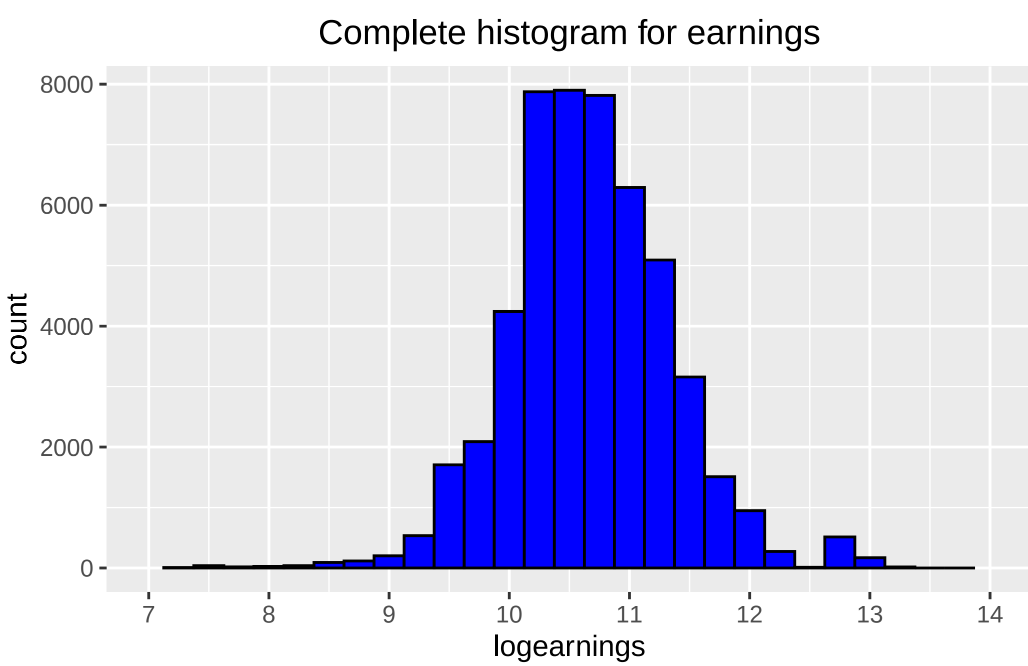
```r
#get a new histogram zooming to the most frequent values

g1<-ggplot(df)+aes(x=logearnings)+geom_histogram(fill="blue", colour="black", stat="bin", binwidth=0.25)+scale_x_continuous(breaks=seq
g2<-ggplot(df)+aes(x=logearnings)+geom_histogram(fill="blue", colour="black", stat="bin", binwidth=0.25)+scale_x_continuous(breaks=seq
g1
```

```
Warning: Removed 2 rows containing missing values or values outside the scale range
(`geom_bar()`).
```




```
g2
```

```
Warning: Removed 53 rows containing non-finite outside the scale range (`stat_bin()`).
Removed 2 rows containing missing values or values outside the scale range
(`geom_bar()`).
```



Splitting Dataset into training and test

We will leave 80% of observations in the training set and 20% in the test set.

```
#set.seed just keeps results random but constant for all using the same seed (so we all will have the same results)
set.seed(1760, sample.kind = "Rejection")
spl = sample(nrow(df),0.8*nrow(df))
head(spl)
```

```
[1] 36155 16660 12408 31822 10816 20591
```

```
# Now lets split our dataset into train and test:
train.df = df[spl,]
test.df = df[-spl,]
dim(df)
```

```
[1] 50742    13
```

```
dim(train.df)
```

```
[1] 40593    13
```

```
dim(test.df)
```

```
[1] 10149    13
```

Making our first regression trees

In order to build a regression tree, we use the function "rpart", as follows:

rpart (formula, data, method="anova", minbucket, cp)

Here are some notes and if method is missing, R tries to make an intelligent guess.

```
# rpart (formula, data, method="anova", minbucket, cp)
# lm(formula, data)
rpart(earnings~ female, data=train.df)
```

```
n= 40593

node), split, n, deviance, yval
      * denotes terminal node

1) root 40593 1.130038e+14 55174.86
  2) female>=0.5 17292 2.290426e+13 44146.22 *
  3) female< 0.5 23301 8.643545e+13 63359.38 *
```

We could compare the leaf values to the means for female or male:

```
(train.df%>%filter(female==1))$earnings%>%mean()
```

```
[1] 44146.22
```

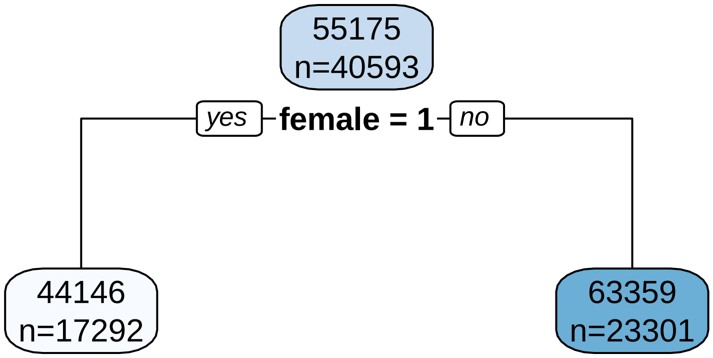
```
(train.df%>%filter(female==0))$earnings%>%mean()
```

```
[1] 63359.38
```

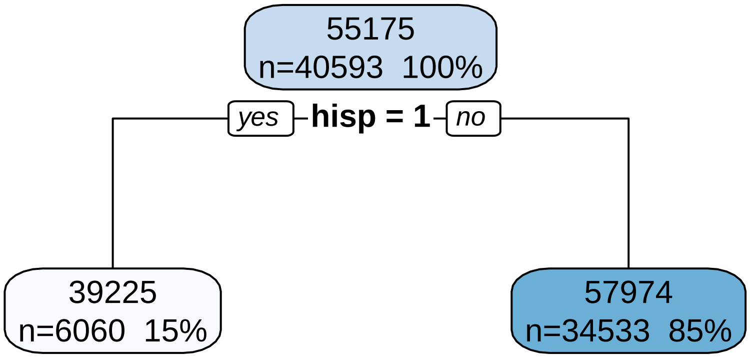
This is all numeric. Let's have a depiction of the tree using rpart.plot()

Regression Tree with female and hispanic

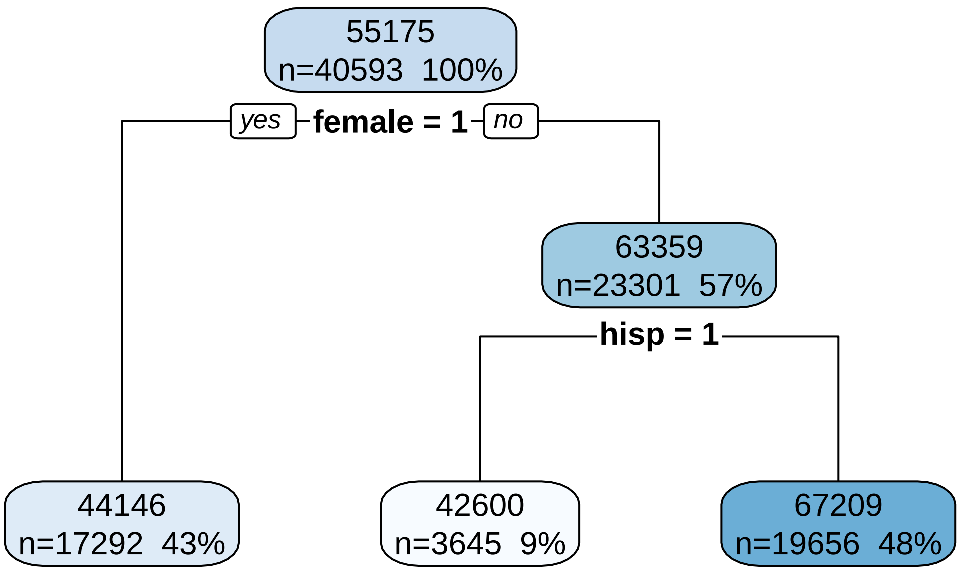
```
tree1<-rpart(earnings~ female, data=train.df)
rpart.plot(tree1,digits=-2,extra=1)   # check out ?rpart.plot for switch info
```




```
tree2<-rpart(earnings~ hisp, data=train.df)
rpart.plot(tree2,digits=-2,extra=101)   # note the 101
```



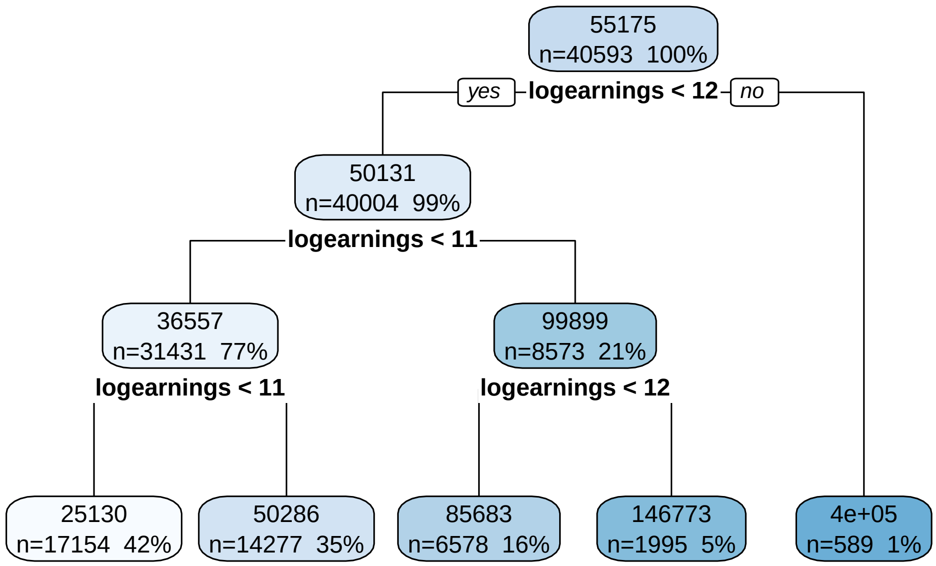

```
tree3<-rpart(earnings~ female+hisp, data=train.df)
rpart.plot(tree3,digits=-2,extra=101)
```



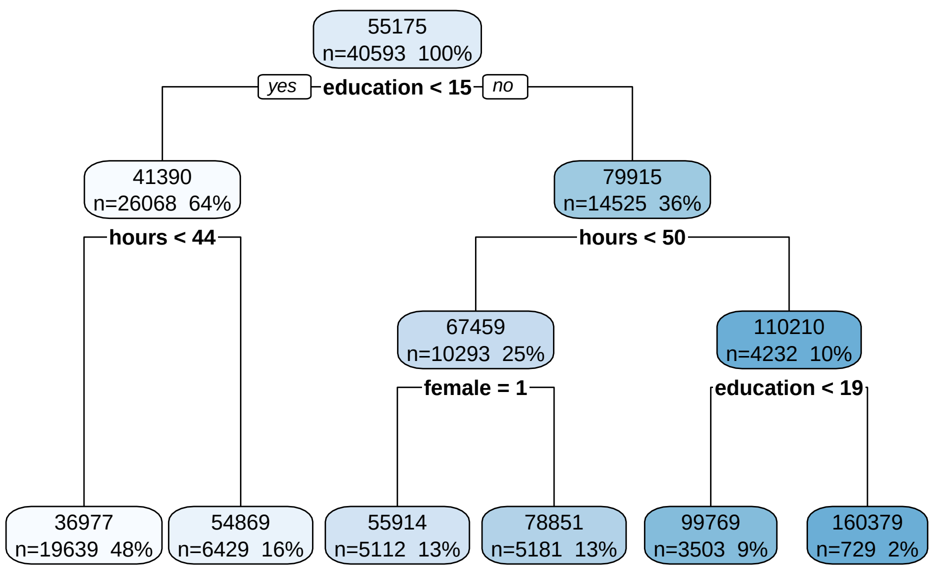

```
# How small could it go?
# Try small values for minbucket=10 and cp=0.00000001 to tree3.
#    Why did it stop?

# add age, so earnings~ female+hisp+age with default minbucket and cp
#    then try minbucket=10 and cp=0.00000001 - thoughts?  cp = 0.001?
```

Regression Tree with all the variables

```
tree4<-rpart(earnings~ ., data=train.df)
rpart.plot(tree4,digits=-2,extra=101)   # try digits = -4
```




```
tree5<-rpart(earnings~ .-logearnings, data=train.df)  # to remove logearinings
rpart.plot(tree5,digits=-2,extra=101)
```

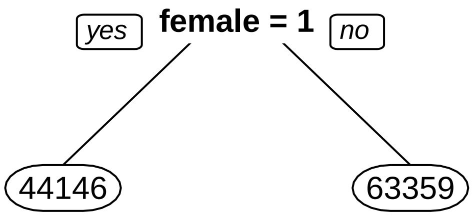



```
# check out the rpart.plot and prp parameters,
#   such as nn=TRUE and box.palette="Red")
```

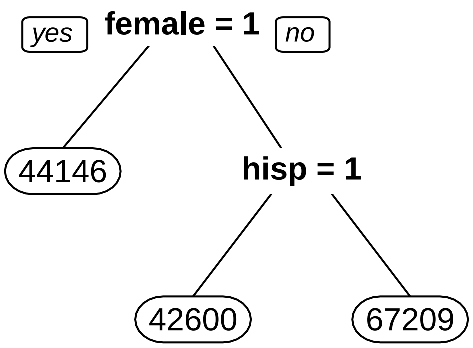
Yes, be sure the output is not part of the input! (e.g., earnings and logearnings)

Although the results are there, they don't look nice. We can print a better tree using "prp":

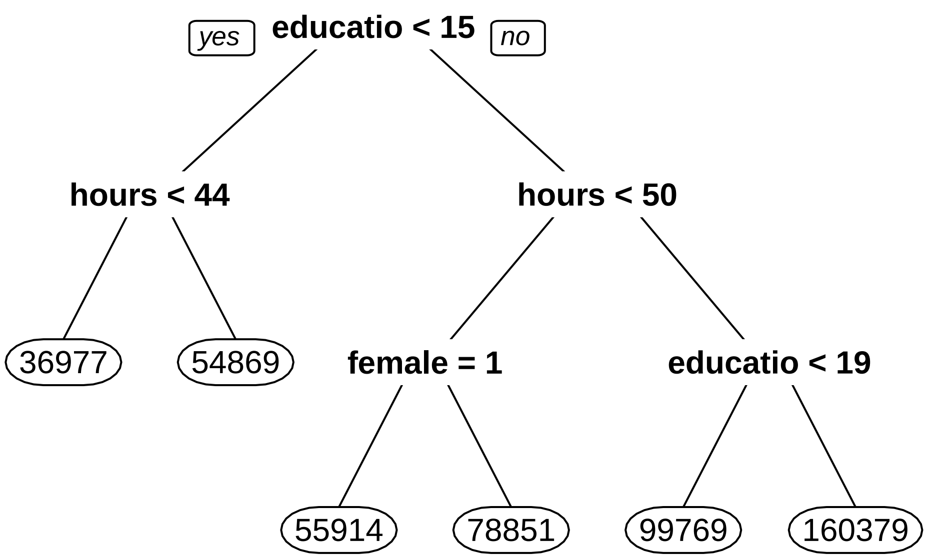
```r
#Try adding negative digits to get results without scientific notation
prp(tree1,digits=-3)
```




```r
prp(tree3,digits=-3)
```

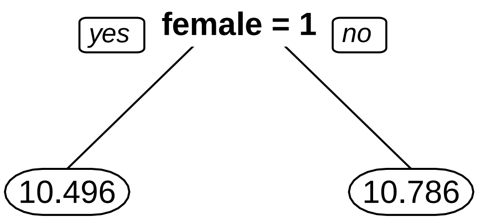



```r
prp(tree5,digits=-3)
```



Let's see how the log earnings work out:

```r
ftreelog<-rpart(logearnings ~ female, data=train.df)
htreelog<-rpart(logearnings ~ hisp, data=train.df)

prp(ftreelog,digits=5)
```




```r
#Check if it is the same value as using just earnings
exp(10.496)
```

```
[1] 36170.53
```

```r
exp(10.786)
```

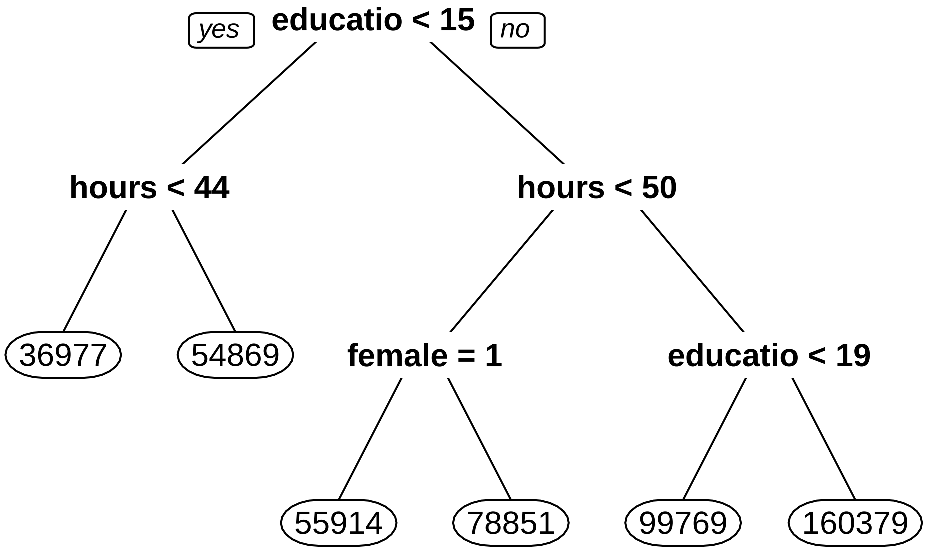
```
[1] 48339.29
```

Lets go back to the more complete tree:

```r
names(df)
```

```
 [1] "earnings"    "age"         "female"      "hisp"        "education"
 [6] "hours"       "week"        "union"       "uncov"       "region"
[11] "race"        "marital"     "logearnings"
```

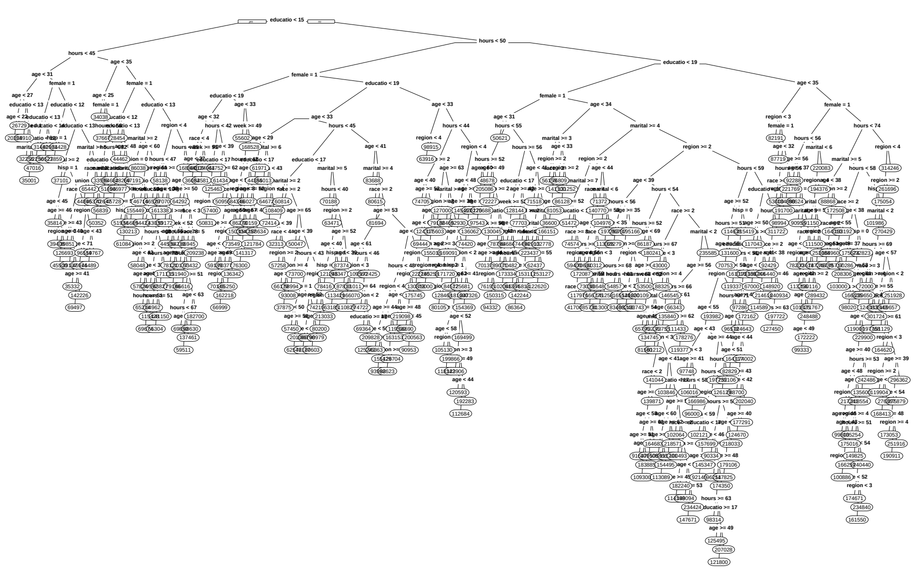
```r
#recall Which variable we cannot use? Use everything else.

basetree<-rpart(earnings ~ .-logearnings,data=train.df)

prp(basetree,digits=-5)
```



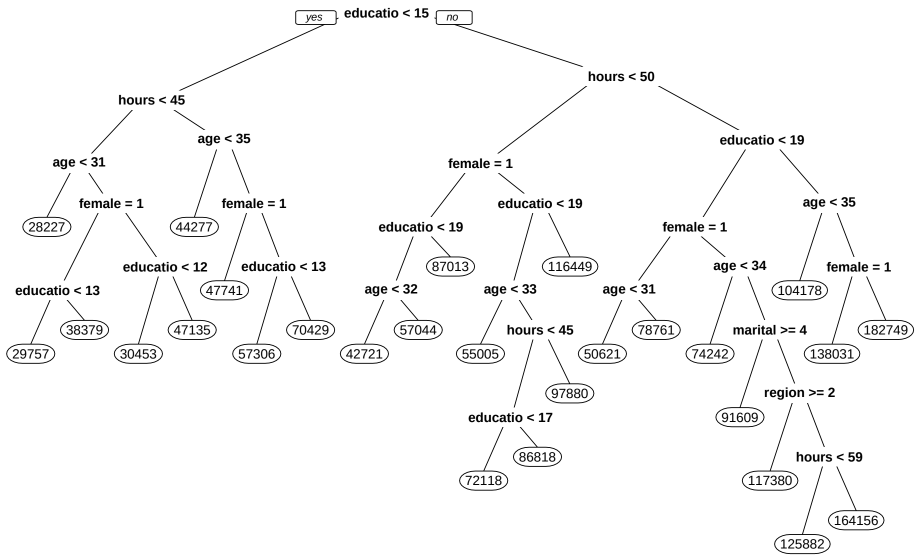
This is actually a nice tree. rpart default control options can lead to quite a decent one. But, let's see what happens if we force it to allow more leaves:

```
#try different cp values to get a bigger tree
prp(rpart(earnings ~ .-logearnings,data=df, method="anova",minbucket=5,cp=0.0001),digits=-5)
```

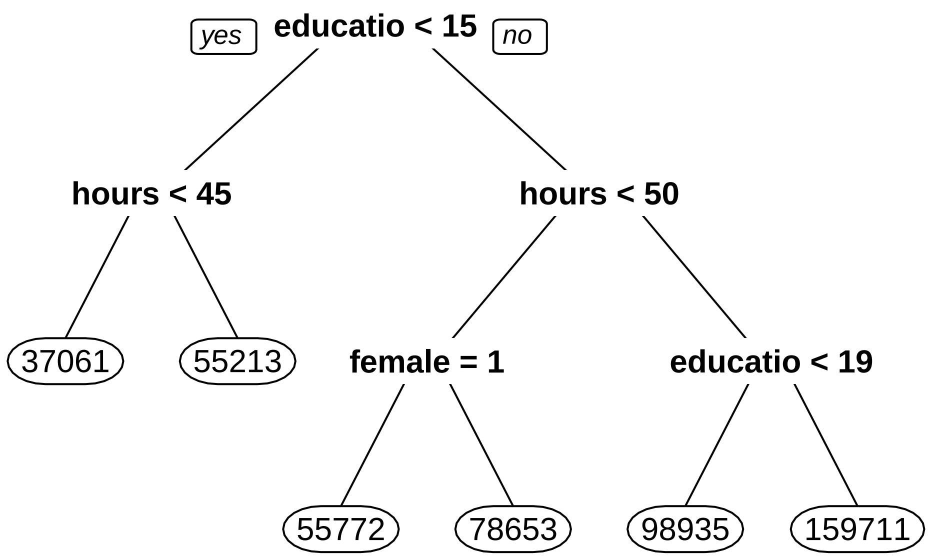
```
Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



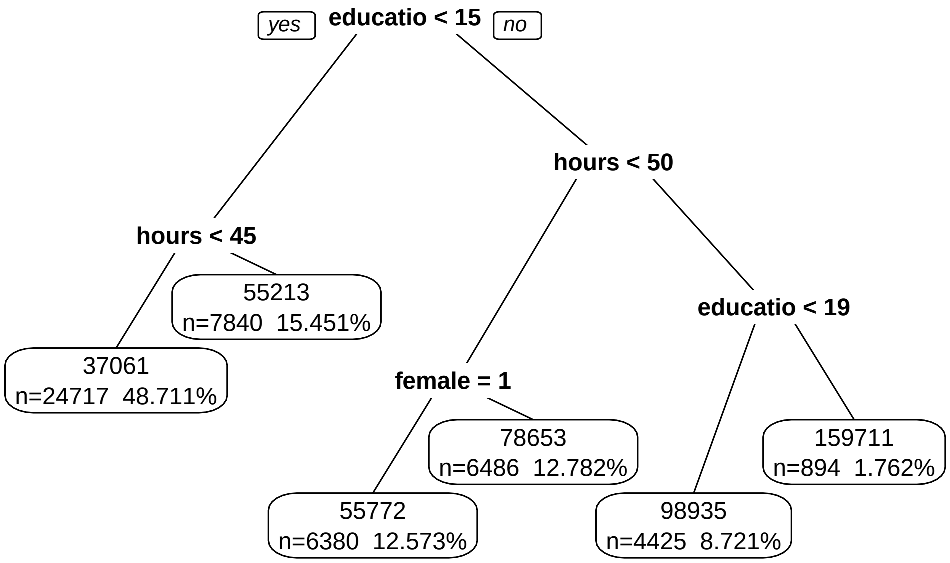
```
prp(rpart(earnings ~ .-logearnings,data=df, method="anova",minbucket=50,cp=0.001),digits=-5)
```




```
prp(rpart(earnings ~ .-logearnings,data=df, method="anova",minbucket=50,cp=0.01),digits=-5)
```



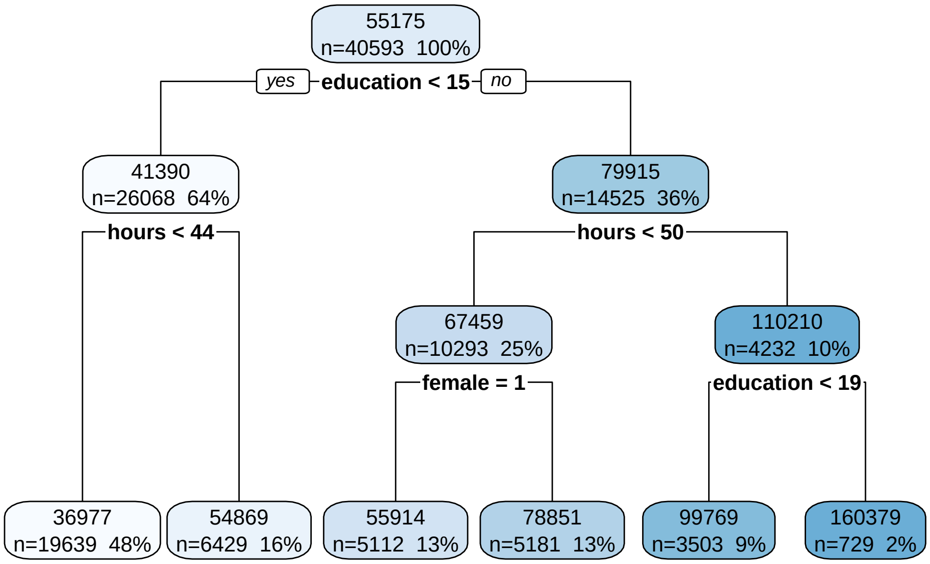

```
prp(rpart(earnings ~ .-logearnings,data=df, method="anova",minbucket=50,cp=0.01),digits=-5,extra=101)
```



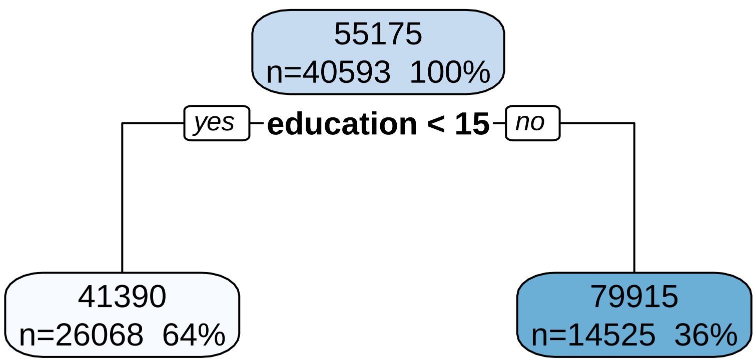

```
# extra = 101 displays observations in each leaf and percentage
```

Cross Validation

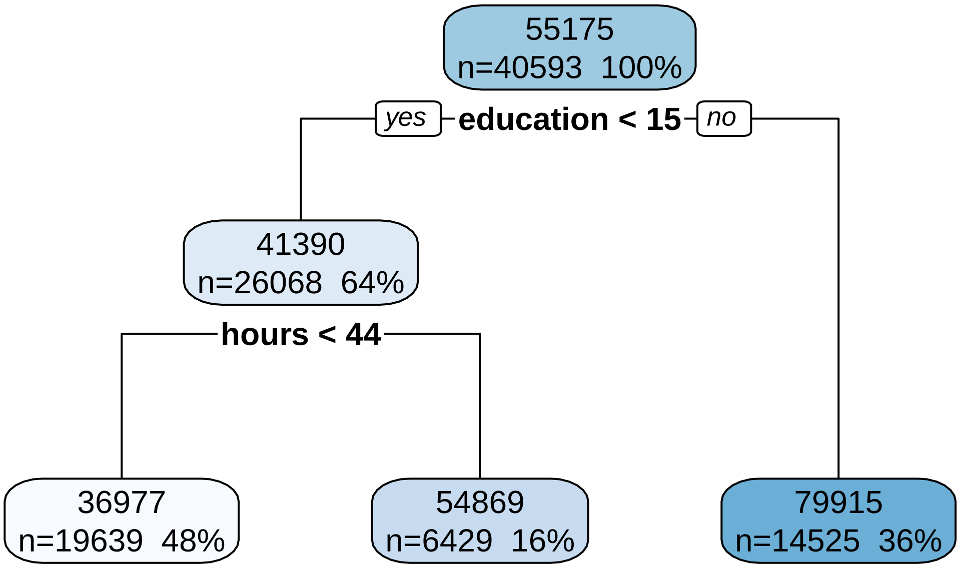
```
set.seed(1760, sample.kind = "Rejection")

#make a tree with a very small value of cp. Not 0 because it will take a long time creating too many splits

tree_cv = rpart(earnings ~ .-logearnings,data=train.df, method="anova")
rpart.plot(tree_cv,digits=-2,extra=101)
```



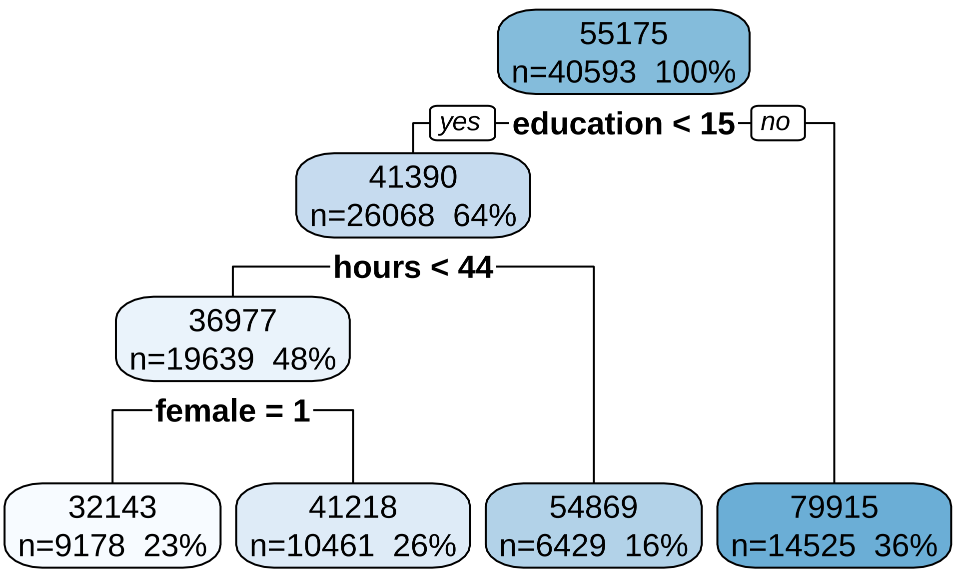

```
tree_cv = rpart(earnings ~ .-logearnings,data=train.df, method="anova",minbucket=5000,cp=0.1)
rpart.plot(tree_cv,digits=-2,extra=101)
```



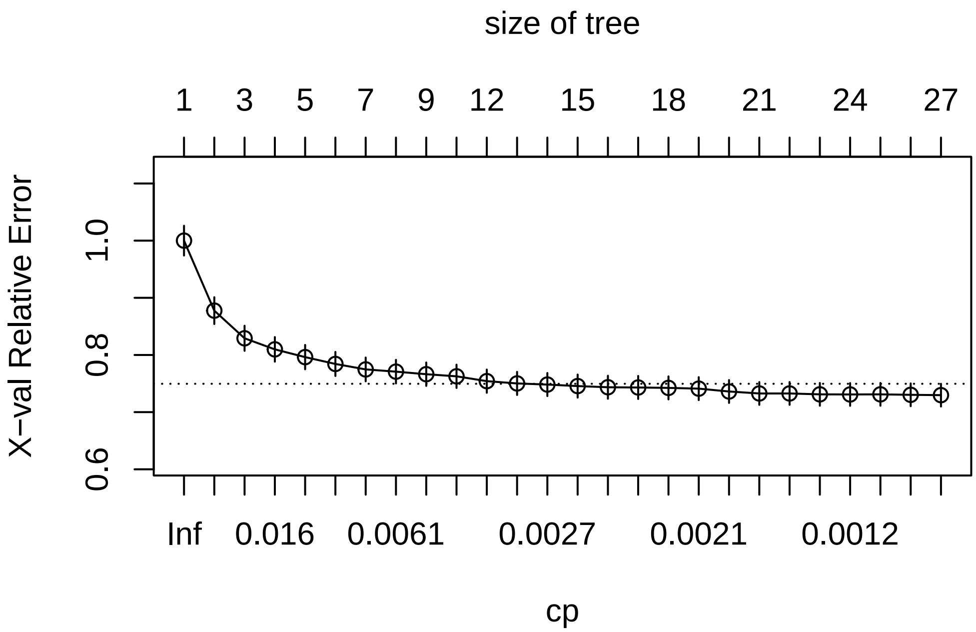

```
tree_cv = rpart(earnings ~ .-logearnings,data=train.df, method="anova",minbucket=5000,cp=0.01)
rpart.plot(tree_cv,digits=-2,extra=101)
```




```
tree_cv = rpart(earnings ~ .-logearnings,data=train.df, method="anova",minbucket=5000,cp=0.001)
rpart.plot(tree_cv,digits=-2,extra=101)
```




```
tree_cv = rpart(earnings ~ .-logearnings,data=train.df, method="anova",minbucket=50,cp=0.001)
plotcp(tree_cv)
```

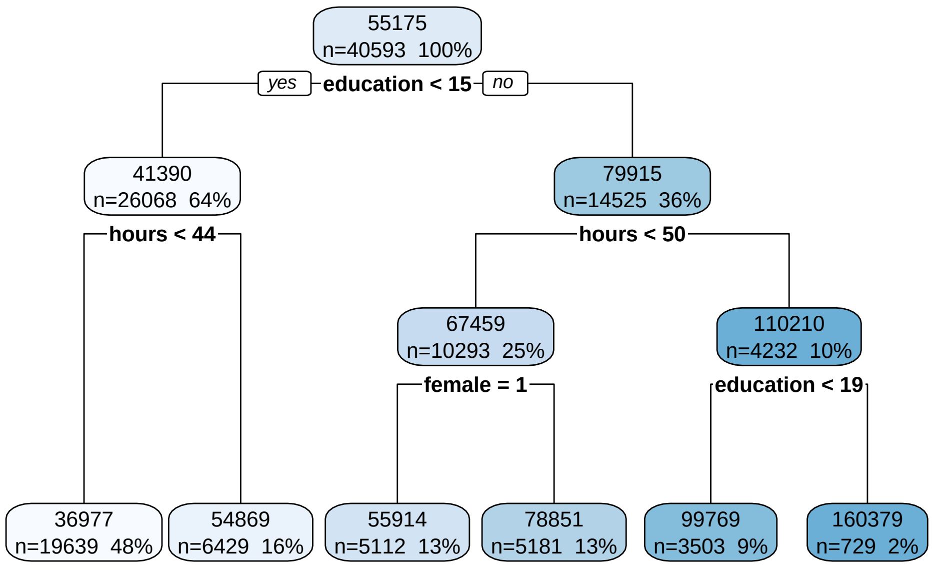



```
#plotcp will give us the relative error in the y axis for a 10-fold cross validation of our dataset, telling us the size of the tree (

#The dotted line in the "plotcp" graph represents the minimum cross-validation error plus one standard deviation. One simple rule of t
```

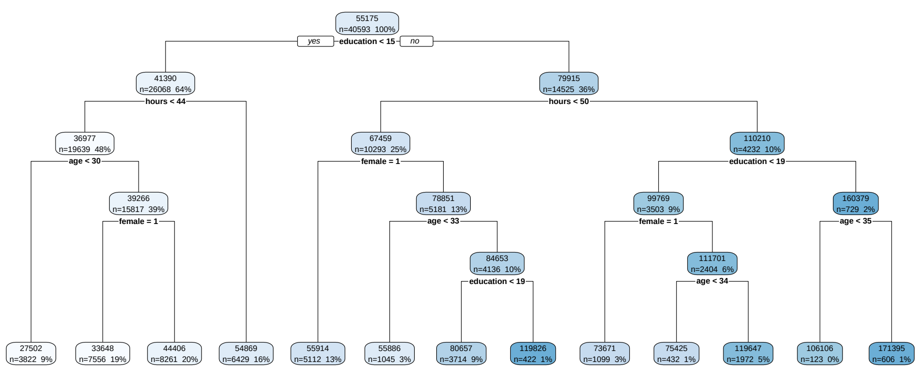
We want to see:

- A tree of size 6 to see what if the model cp is too large (cp = 0.01)
- A tree of size 13 based on the ref. line, so we have to use cp of about 0.003
- A tree of size of ~ 24 (modeler chosen desired size), so we have to use cp 0.00125

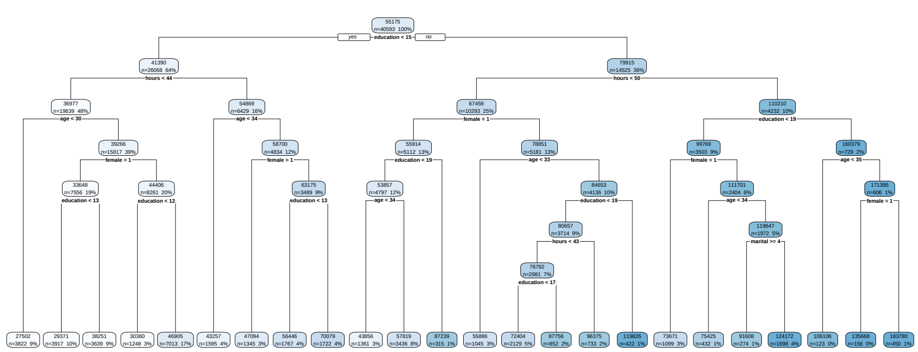
```
tree01 = rpart(earnings ~ .-logearnings,data=train.df, method="anova",minbucket=50,cp=0.01)
rpart.plot(tree01,digits=-2,extra=101)
```




```r
treea = rpart(earnings ~ .-logearnings,data=train.df, method="anova",minbucket=50,cp=0.003)
rpart.plot(treea,digits=-2,extra=101)
```




```r
treeb = rpart(earnings ~ .-logearnings,data=train.df, method="anova",minbucket=50,cp=0.00125)
rpart.plot(treeb,digits=-2,extra=101)
```



Lets get predictions for both models:

```
test.df$pred01 = predict(tree01, newdata= test.df)

test.df$preda = predict(treea, newdata= test.df)

test.df$predb = predict(treeb, newdata= test.df)

head(test.df)%>%relocate(preda,predb, pred01)
```

```
      preda    predb   pred01 earnings age female hisp education hours week
4  33647.75 38251.32 36977.02    47000  41      1    0        13    40   52
6  33647.75 38251.32 36977.02    33000  66      1    0        13    40   52
26 33647.75 38251.32 36977.02    71000  33      1    0        14    40   52
30 44405.63 46905.08 36977.02    26000  52      0    0        14    40   52
32 44405.63 46905.08 36977.02    21840  37      0    0        13    40   52
35 44405.63 46905.08 36977.02    50000  43      0    0        12    40   52
   union uncov region race marital logearnings
4      0     0      1    1       1   10.757903
6      0     0      1    1       5   10.404263
26     0     0      1    1       1   11.170435
30     0     0      1    1       1   10.165852
32     0     0      1    1       1    9.991498
35     0     0      1    1       1   10.819778
```

Which model is better? To get out of sample R square:

```
mean_train = mean(train.df$earnings)  #grab the mean for calc below

# Then, we compute the sum of squared errors (SSE) using our tree:

SSE01 = sum((test.df$earnings - test.df$pred01)^2)
SSEa = sum((test.df$earnings - test.df$preda)^2)
SSEb = sum((test.df$earnings - test.df$predb)^2)

SSE01
```

```
[1] 1.989891e+13
```

```
SSEa
```

```
[1] 1.895557e+13
```

```r
SSEb
```

```
[1] 1.842814e+13
```

```r
print(paste("Tree CP=0.01 has a SSE of", SSE01))
```

```
[1] "Tree CP=0.01 has a SSE of 19898908525677.2"
```

```r
print(paste("Tree A has a SSE of", SSEa))
```

```
[1] "Tree A has a SSE of 18955573723196.2"
```

```r
print(paste("Tree B has a SSE of", SSEb))
```

```
[1] "Tree B has a SSE of 18428139497170"
```

```r
# And the total sum of squared errors (SST) using our simple benchmark model
# (the mean in the training set)

SST = sum((test.df$earnings - mean_train)^2)

# With that, we finally get

OSR2.01 = 1 - SSE01/SST
OSR2a = 1 - SSEa/SST
OSR2b = 1 - SSEb/SST

OSR2.01
```

```
[1] 0.2157945
```

```r
OSR2a
```

```
[1] 0.2529708
```

```r
OSR2b
```

```
[1] 0.2737567
```

Let's see the MAE for comparisons:

```r
MAE.01 = mean(abs(test.df$earnings - test.df$pred01))
MAEa = mean(abs(test.df$earnings - test.df$preda))
MAEb = mean(abs(test.df$earnings - test.df$predb))



MAE.01
```

```
[1] 24780.85
```

```r
MAEa
```

```
[1] 23928.23
```

```r
MAEb
```

```
[1] 23307.01
```