Senate Judiciary Committee Report: An Analysis of Federal Capital Charges with CART

Michael Harrison Lee

Tuesday, March 20, 2018

Introduction:

The United States death penalty has long been controversial due to the alleged influence of "illegitimate factors" such as race, nationality, gender, ethnicity, and disability in the decision-making process. In an ideal world, the severity of punishment for a crime would be based solely on the severity of said crime; however, in reality human biases and systemic inefficiencies may prevent this from being the case. To draw clues as to whether or not the system as it stands is "fair," we might ask: to what degree can we use these factors such as race, nationality, gender, and ethnicity to successfully classify capital charges in the federal criminal justice system? That is, how much of the decision on whether or not to issue a capital charge can be explained by variables representing these "illegitimate factors?"

To answer this question, we will be examining a dataset consisting of criminal cases in the federal justice system during the Clinton Administration. The data were collected in an attempt to consider capital charging practices and include all homicide cases where it was legally permitted to seek the death penalty. The CART algorithm, more commonly known as the "decision tree," will be used to partition and understand the data.

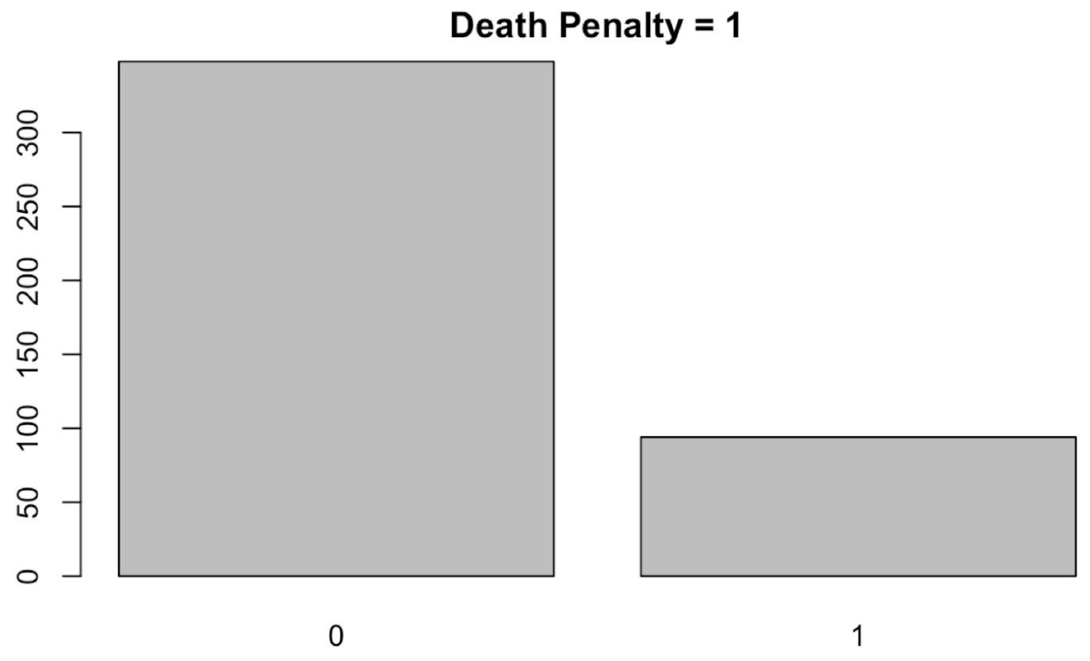
Data Preparation:

All variables in the raw dataset are coded as numerical, despite most being categorical in nature. In order for our tree-based analysis to work properly, we must convert all of these into factors with the exception of numbervictim (the number of homicide victims per case), which can remain as numeric. In addition, there is a duplicate column for bkplead (victim plead for their life before the killing) which must me removed, and the labeled levels for working(defendant's employment status) must be changed to 0 & 1 to match the data. Lastly, even though there are many missing values, tree based methods are generally considered to be adept at dealing with these. Because of this, we will only remove rows for which there is no value for the response (death).

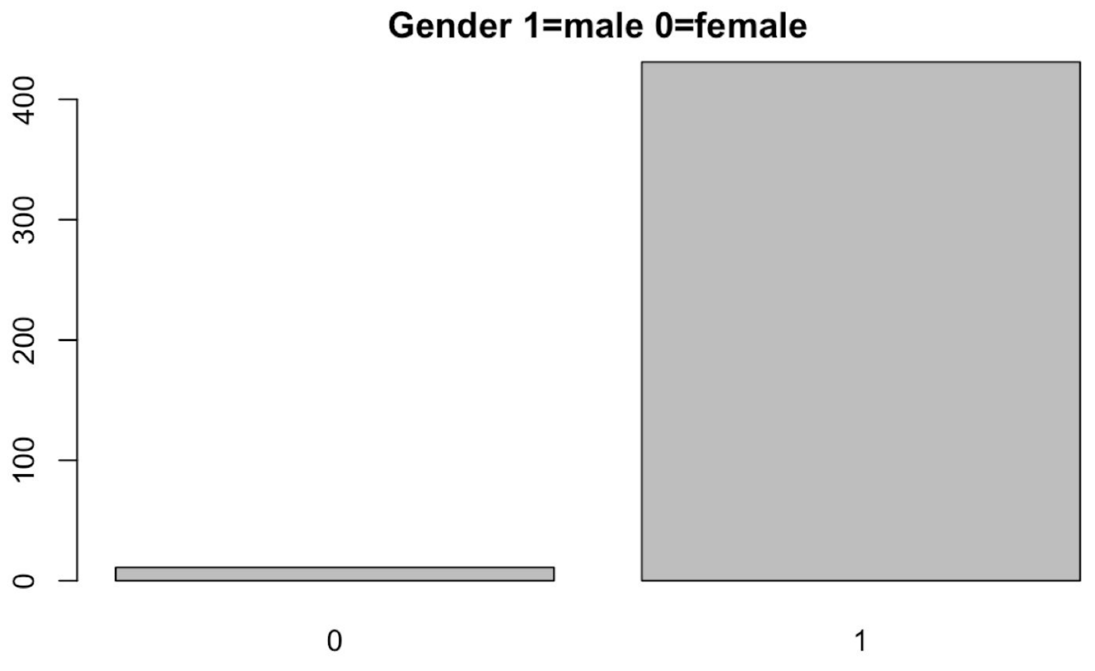
Because we will be looking to assess the relationship of "illegitimate factors" with decisions to issue the death penalty, we will define illegitimate factors in the dataset as the defendant's gender, defendant's race (white, black, hisp), defendant's education level, defendant's birthplace, gender of the victim (vmale), and race of the victim (vwhite, vblack, vhisp).

Univariate Statistics:

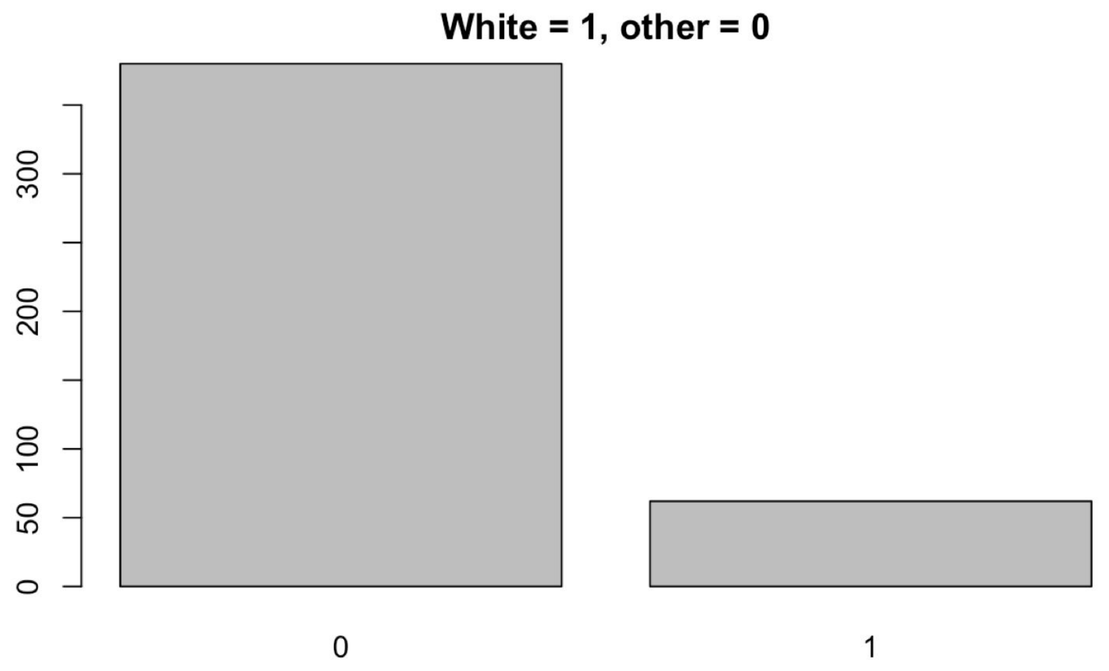
Now that we have cleaned and prepared our dataset, let us take a look at the univariate distributions of observations among variables.



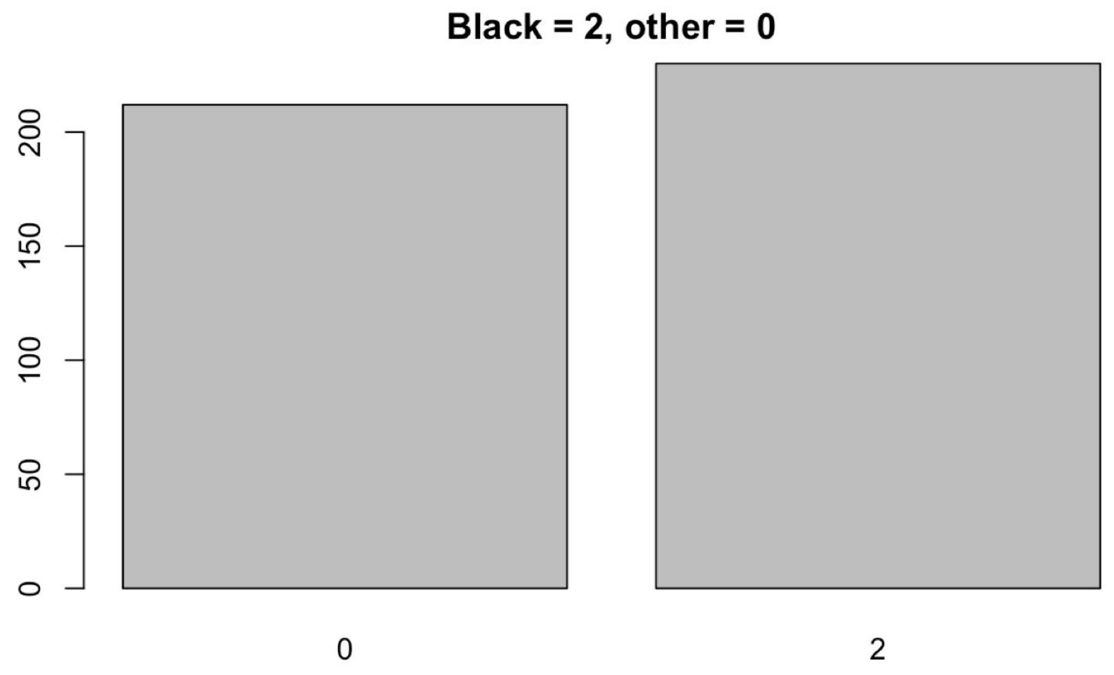
Starting with the response, we observe that a significant majority of cases in which it is legal to seek the death penalty do not result in a capital charge.



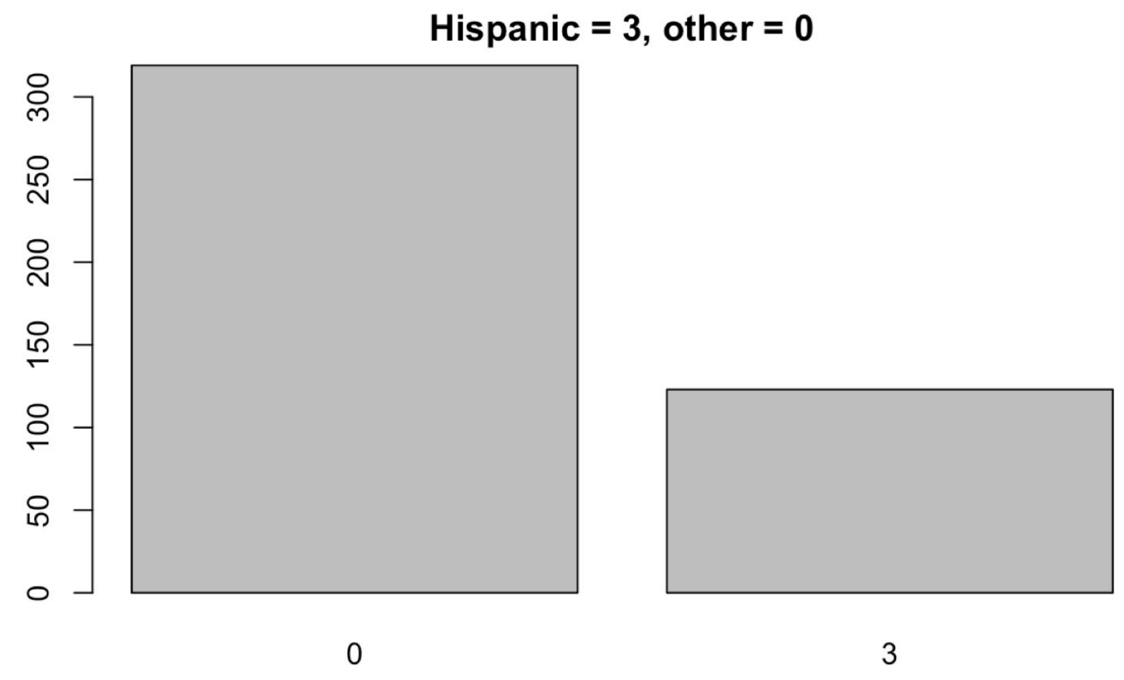
With regards to gender, the vast majority of potential capital offenders are male.



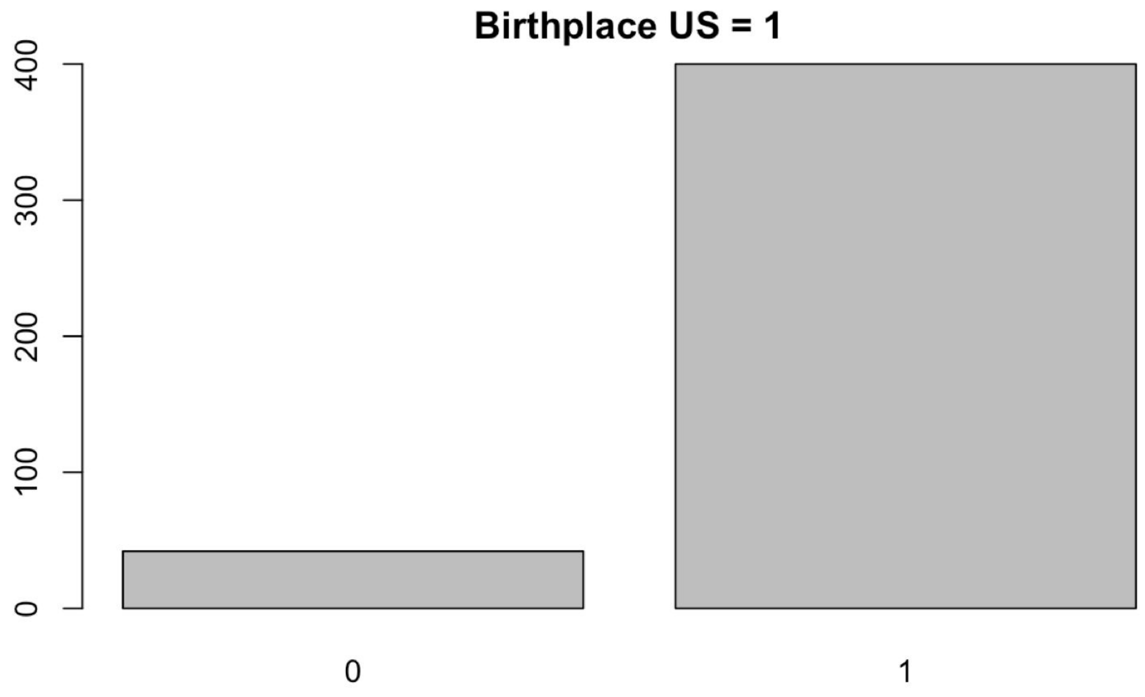
Shifting our attention to race, a relatively small proportion of defendants are White.



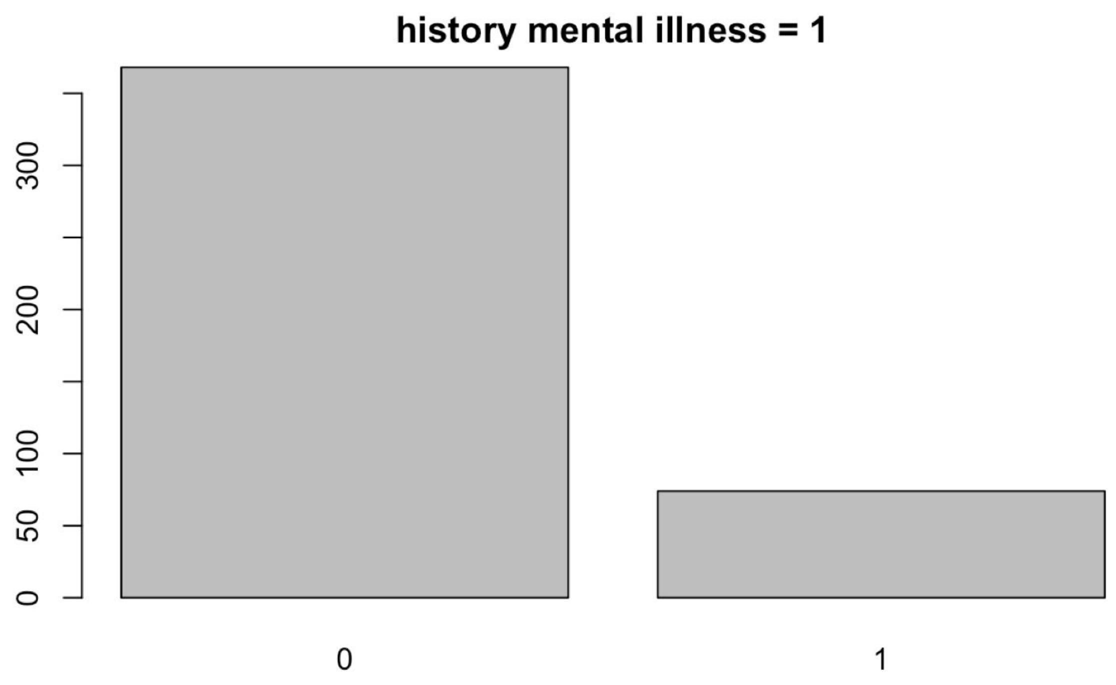
A little more than half of defendants are Black.



Roughly one third of defendants are Hispanic.

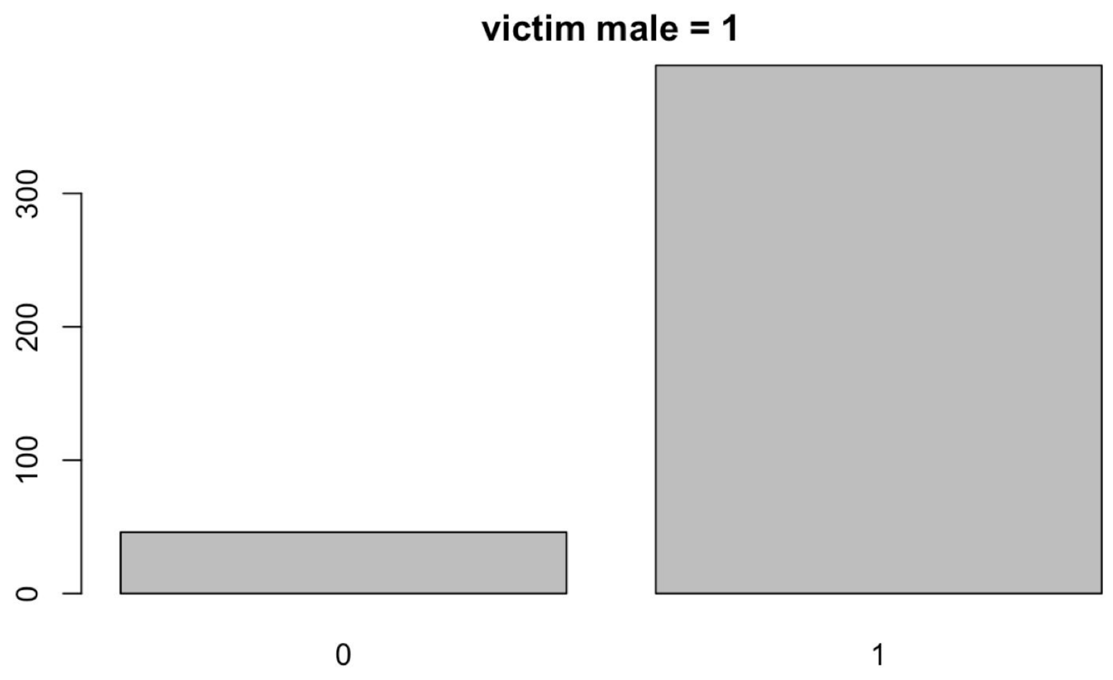


With regards to birthplace. The majority of defendants eligible for capital charges are born in the United States.

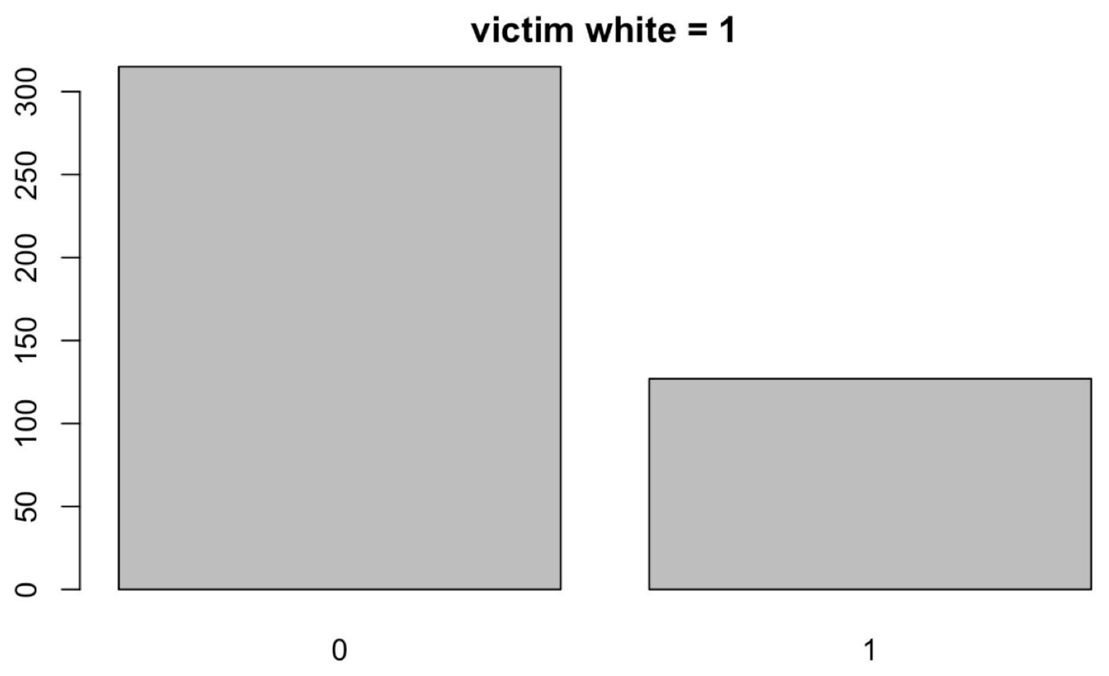


A small, but significant portion of defendants have a history of mental illness.

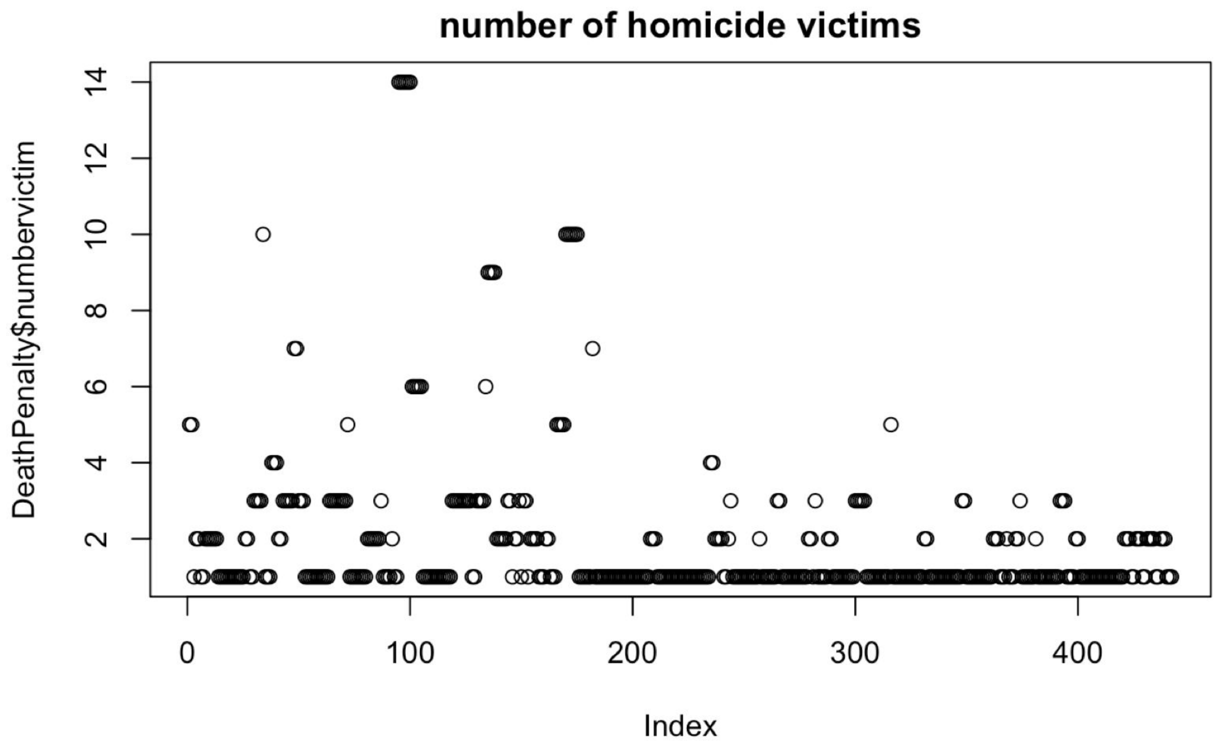
Now, let us turn our attention to victims:



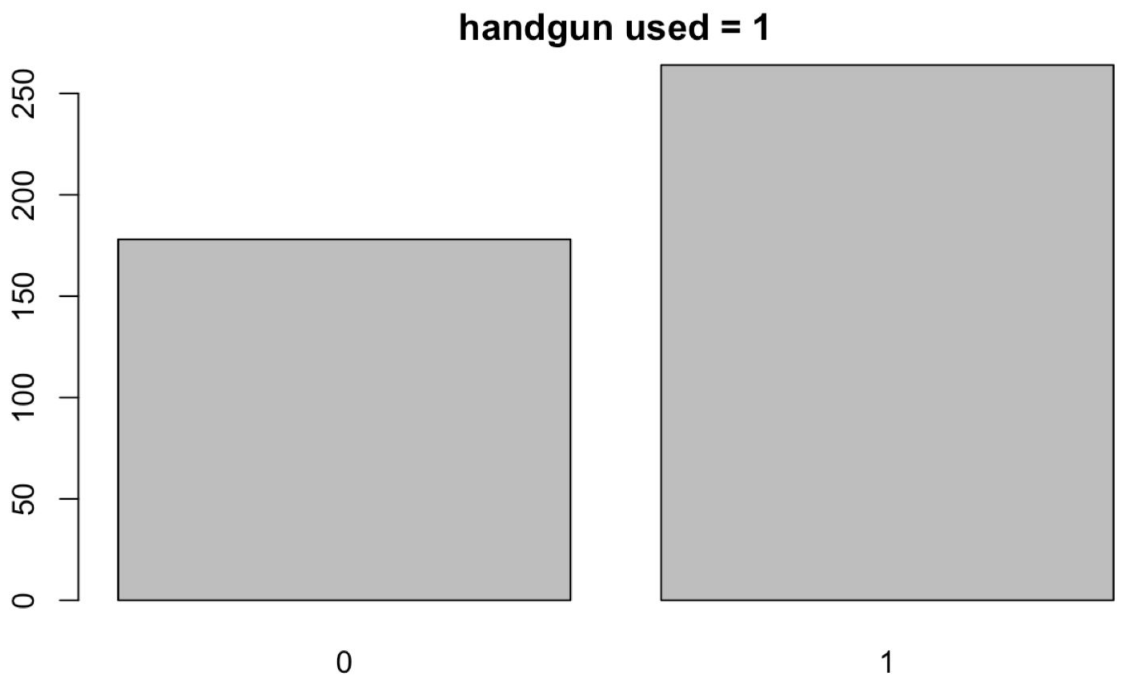
The vast majority of victims are male.



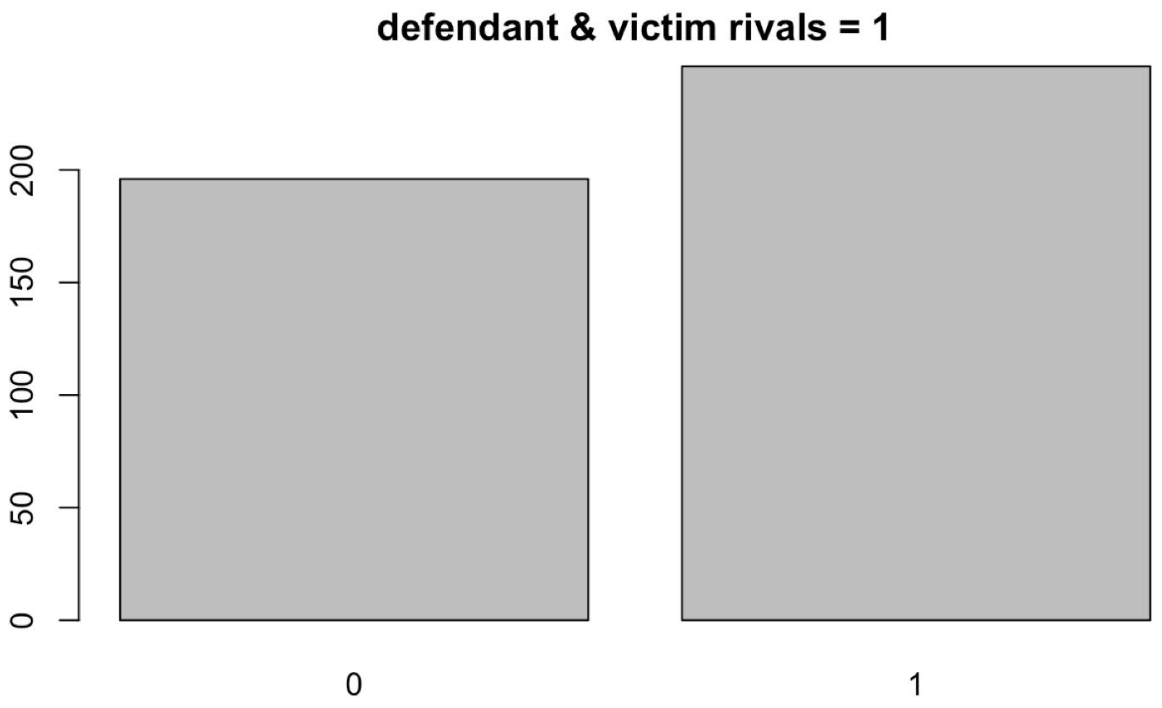
Most victims are non-whites.



Most cases had 1 or 2 victims.



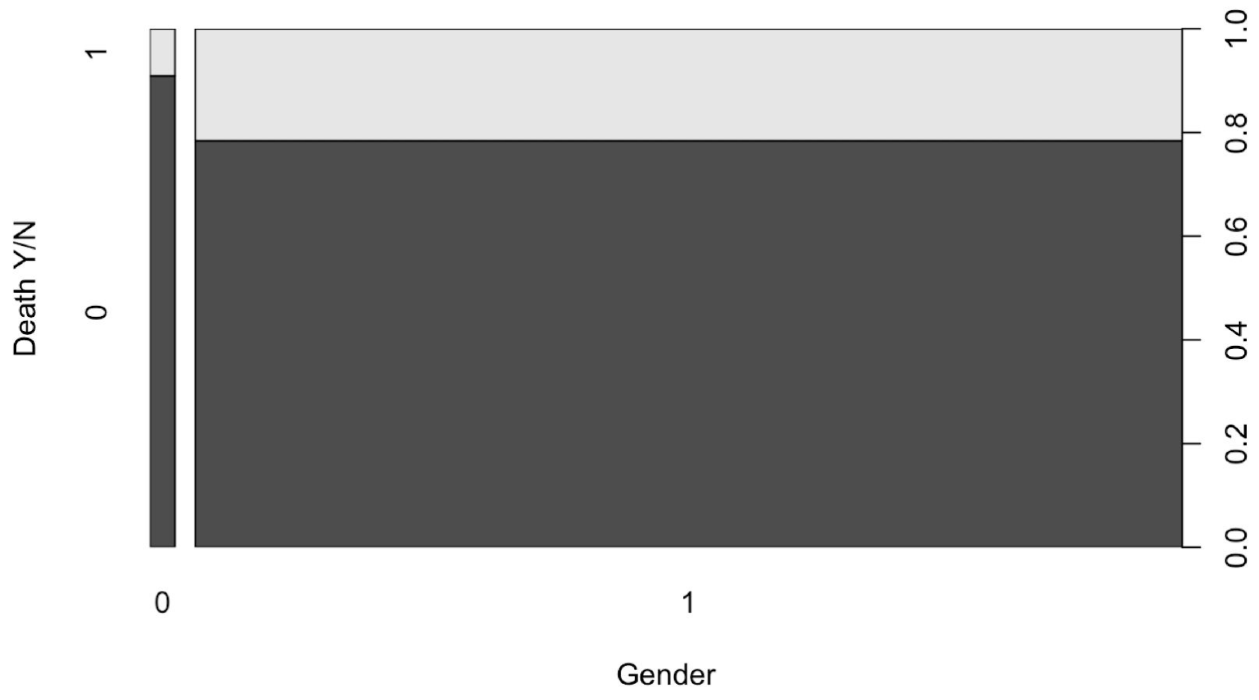
Most of the offenses were committed with a handgun.



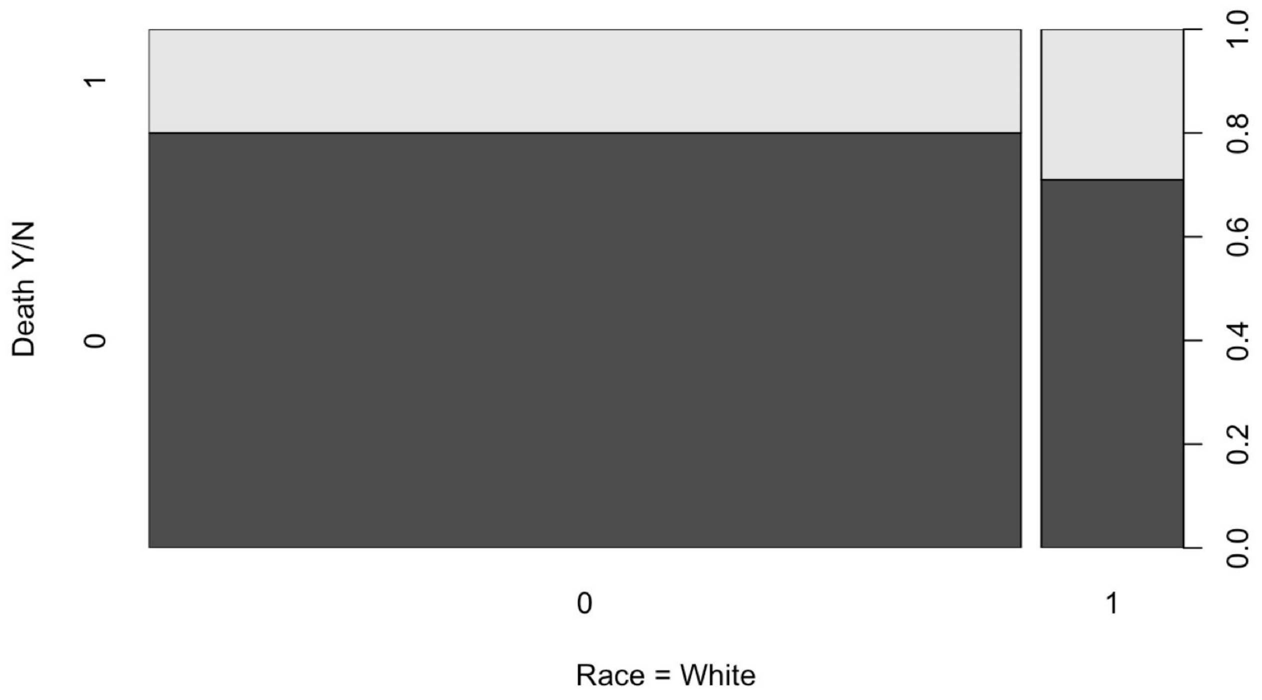
Most of the time, the defendant knew the victim and there was known animosity between the two.

Bivariate Statistics:

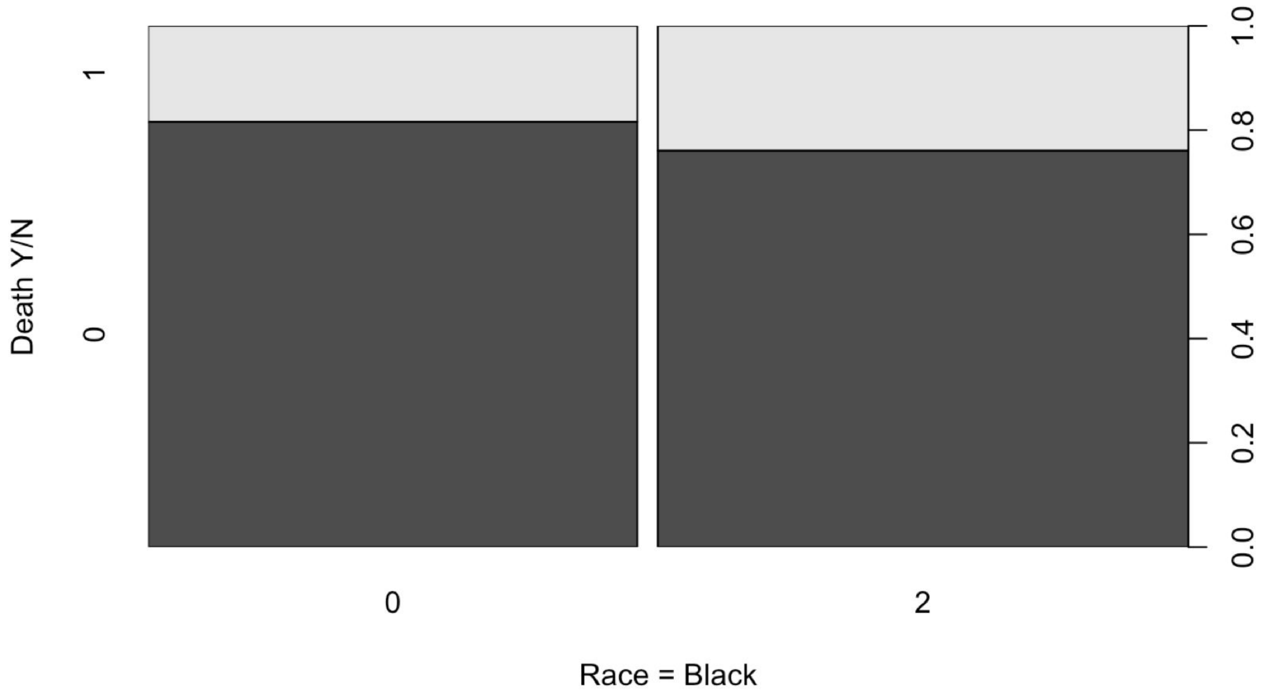
Because our analysis is intended to investigate the relationship of "illegitimate factors" with the death penalty verdicts, we will examine the bivariate relationships of gender, race of the defendant (white, black, hisp), education, birthplace, gender of the victim (vmale), and race of the victim (vwhite, vblack, vhisp) with the response (death).



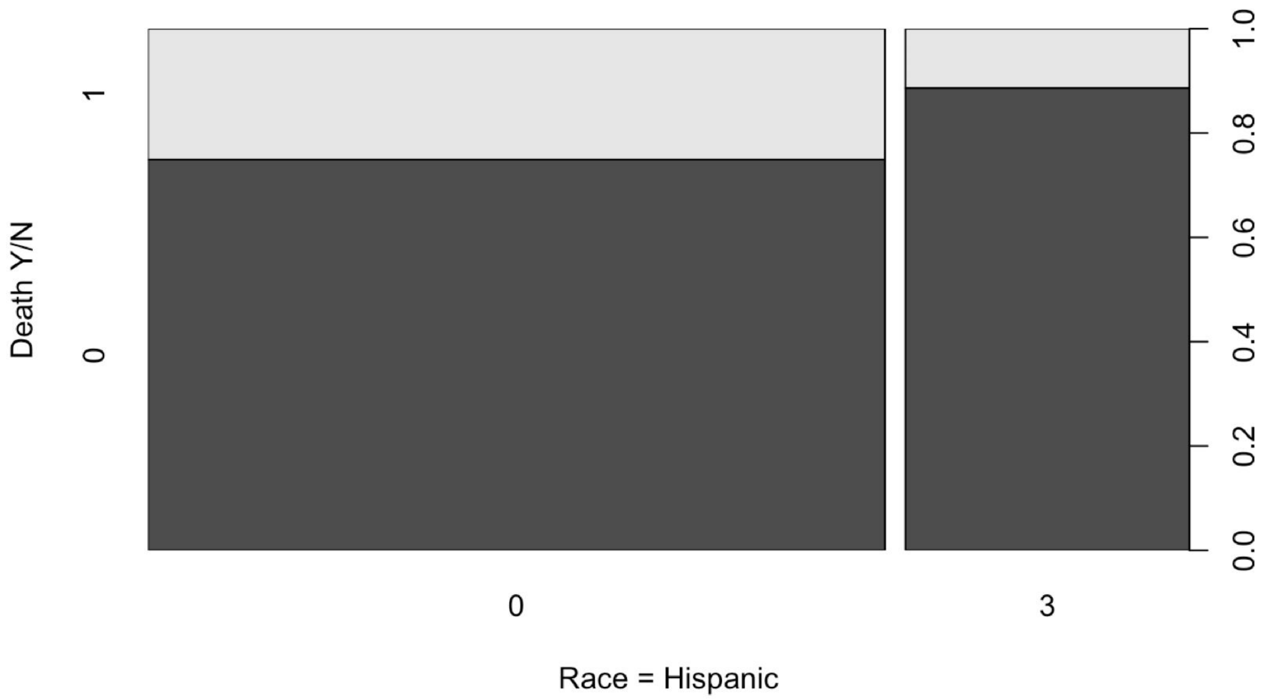
Starting again with gender, we can see that women are slightly more likely to be sentenced to death than men.



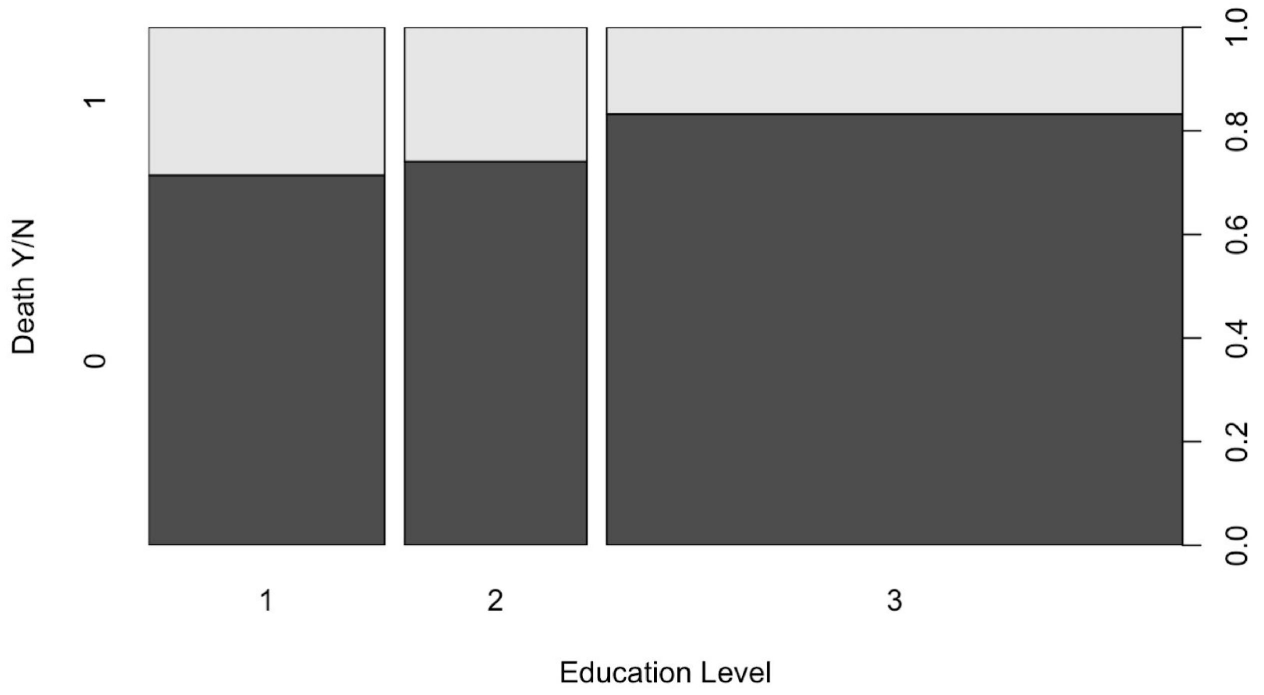
Whites are less likely to be sentenced to death.



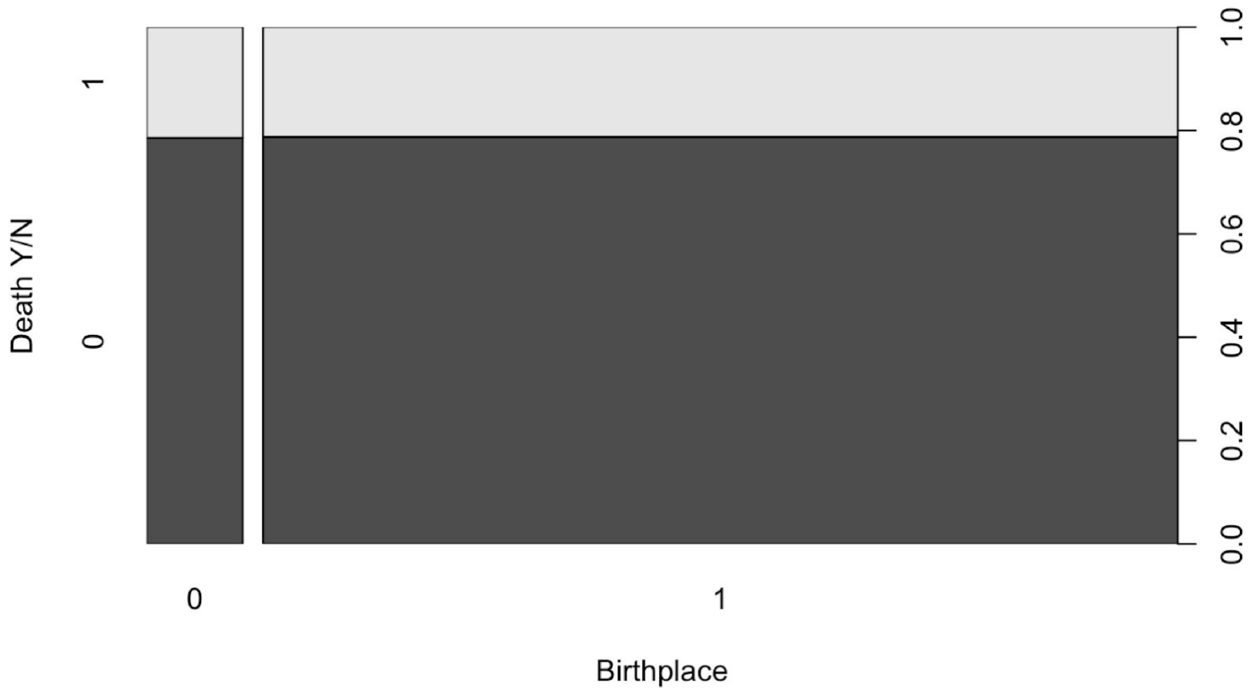
Blacks are very slightly less likely to be sentenced to death.



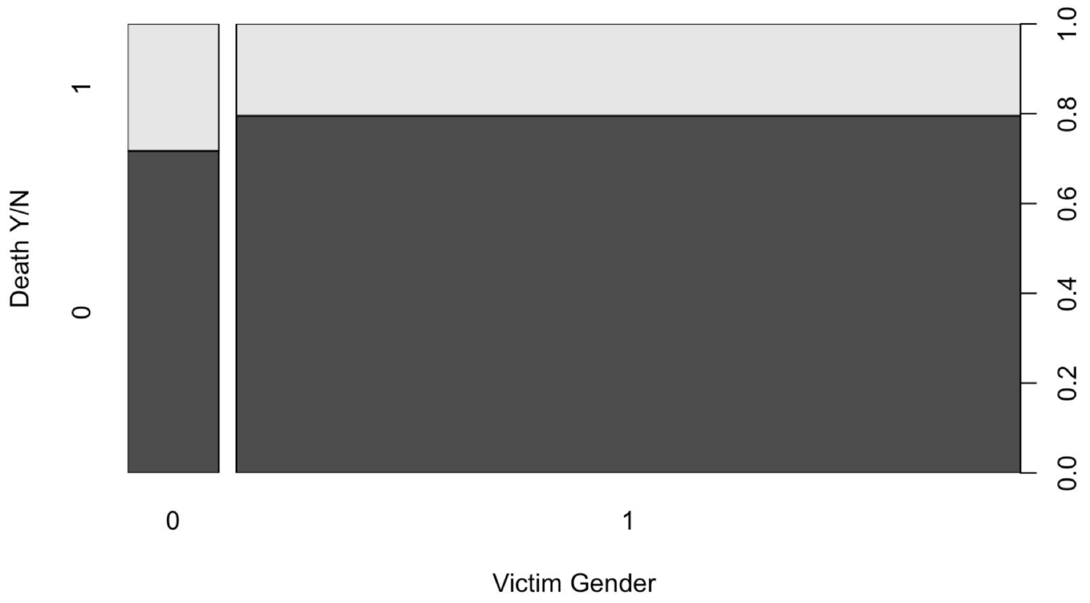
Hispanics are more likely to be sentenced to death.



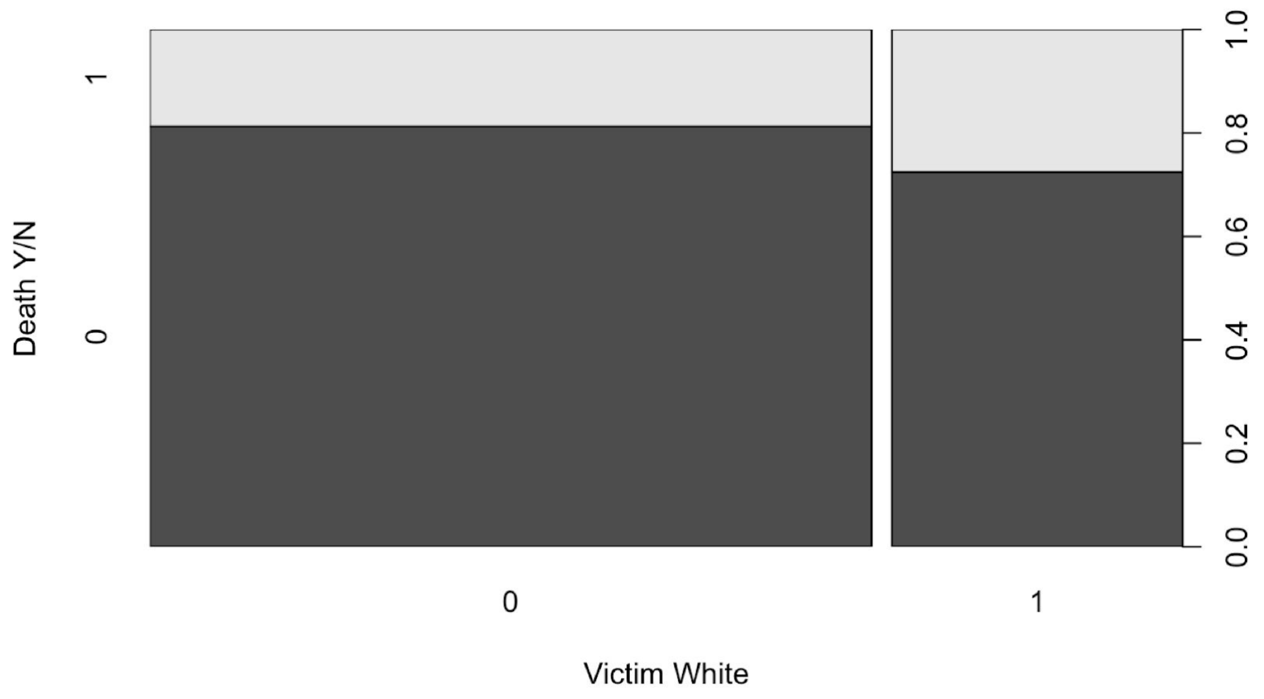
Ignoring unknowns, those who did not graduate high-school are more likely to receive a capital charge.



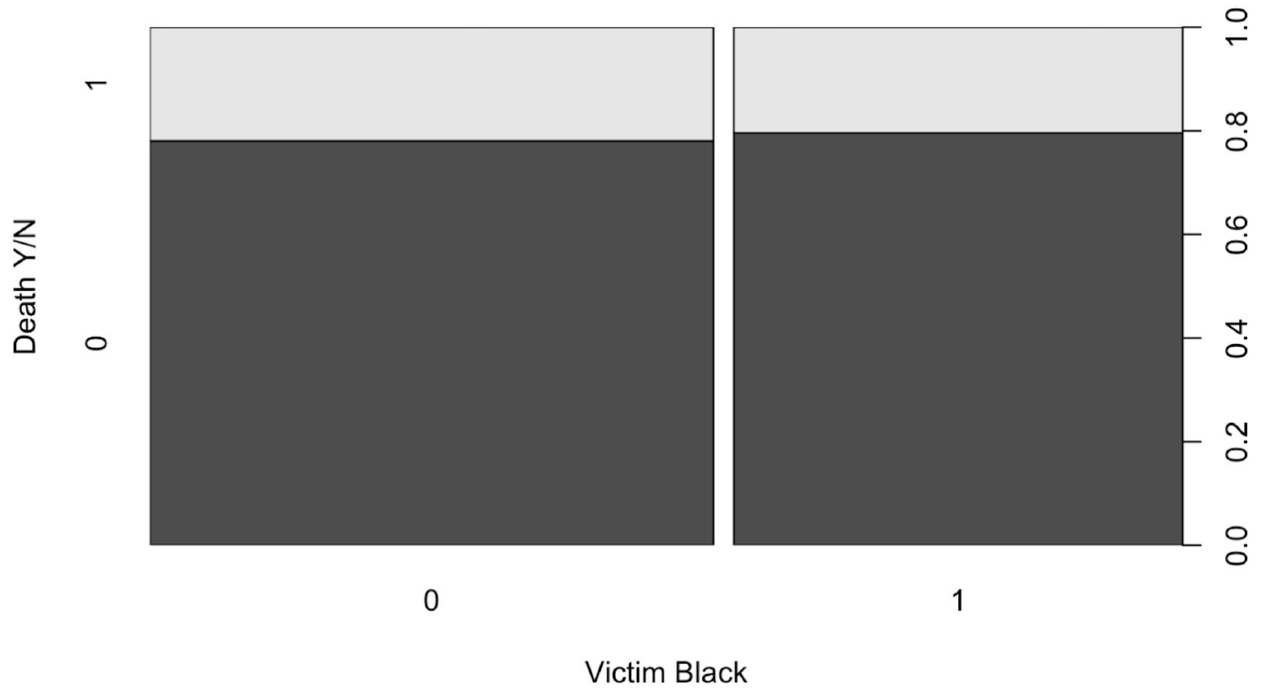
Those born in the United States are sentenced to capital punishment at roughly the same rate as those born outside.



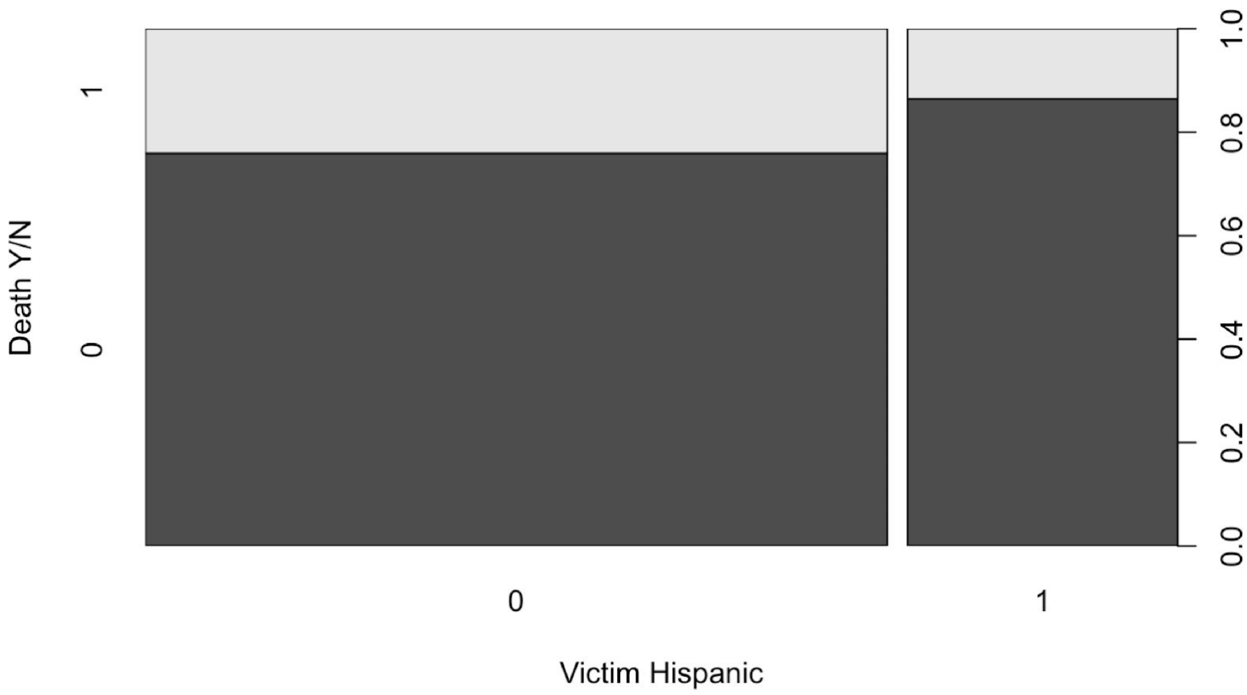
The defendant is more likely to be sentenced to death if the victim is male.



The defendant is less likely to be sentenced to death if the victim is white.



Whether or not the victim is black appears to have little correlation to capital charges.



The death penalty is more likely to be issued when the victim is hispanic.

Assumptions for Analysis:

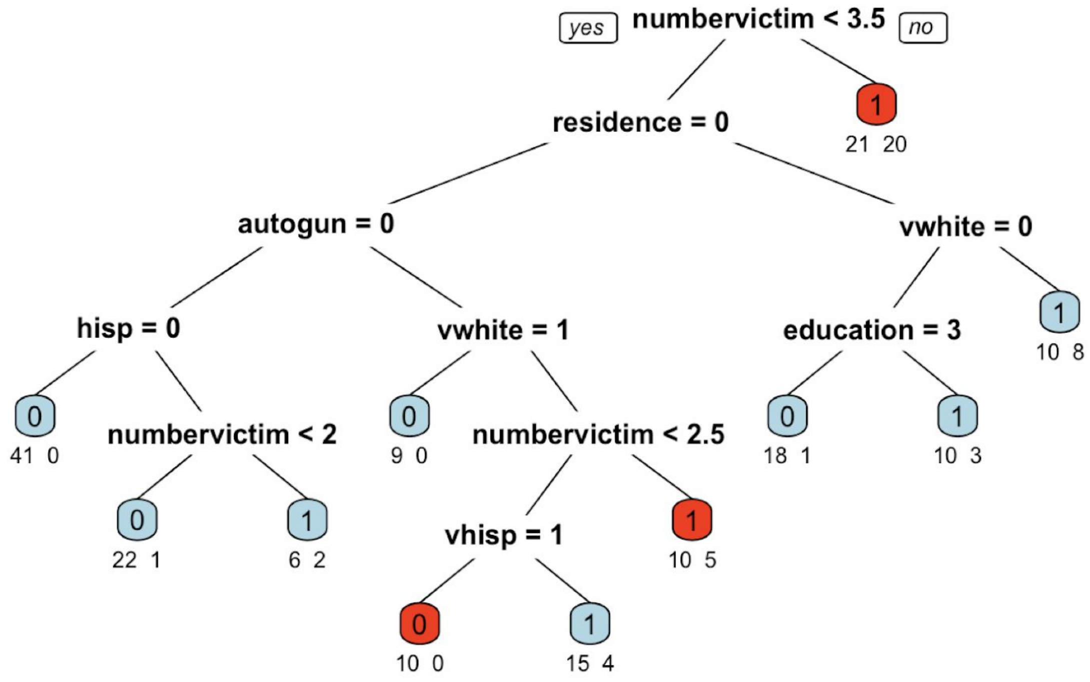
Because the analysis will be used to draw conclusions in forward-looking policy adjustments, we will be working at Level 2. That is, we will be attempting to grow a classification tree which best approximates the underlying distribution of capital charges conditional on various factors. Our estimation target in this case is an explicit, tree based, approximation of the true response surface. Because the data contain all cases within the federal system in which it was legal to seek the death penalty during the Clinton Administration, it is reasonable to assume that the data were generated in a way that is representative of the natural, joint distribution underlying the desired response. While there may have been some policy changes from the Clinton Administration to present day, we will assume they have not been significant enough to alter the underlying probability mass function.

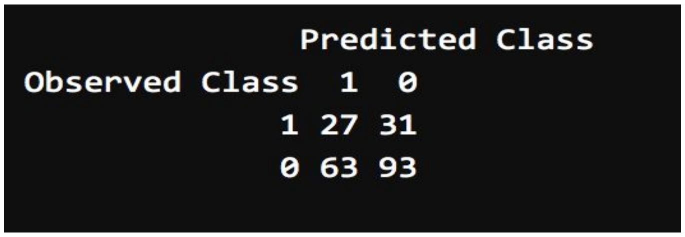
Given that CART "data snoops" to form partitions, it is a high variance algorithm. To address the problem of overfitting, we will divide the data into training, evaluation, and test sets by randomly sampling the row indices. This will allow us to fit our model repeatedly, tune its performance on a random realization of the dataset, and perform our analysis by dropping the uncompromised test data down the resulting tree. Node complexity will be managed by tuning the "cp" parameter, and prior probabilities will calculated and set in advance.

CART

The empirical prior probability distribution is 0.2321981 for "death" and 0.7678019 for "no-death." For this analysis, we will assume that a false positive (predict "death", observe "no death" is 2 times as costly as a false negative (predict "no-death", observe "death"). Adjusting and normalizing these values to our desired cost ratio, we end up with a prior probability of 0.3768844 for "death" and 0.6231156 for "no death." To pre-prune the tree, we will grow trees on the training set with complexity parameter values of 0.01, 0.02, 0.03, 0.04, 0.05 and examine the performance of each one on the evaluation set using overall misclassification rate and qualitative judgements of tree structure and confusion matrix performance. Using this methodology, we found that a cp value of 0.01 gave the best performance.

Our output on the test set is:





```
                Predicted Class
Observed Class   1  0
               1 27 31
               0 63 93
```

	Test	Bootstrap SE
Overall Misclassification Rate	0.4392523	0.02324267
False Positive Use Error	0.7	0.02458195
False Negative Use Error	0.25	0.02214529
False Positive Model Error	0.4038462	0.02348046
False Positive Model Error	0.5344828	0.03446733

Interpretation of the Results:

Looking at our tree we can first see that a victim count of over 3.5 on average is slightly correlated with the decision to issue a capital charge, although this terminal node is not homogenous to make a definite claim in this regard. Given a "low" victim count and that the homicide occurred in the victim's home, the death penalty was more likely to be issued if the victim was white. However, this is a relatively weak association once again. What is more surprising is that, conditional on the homicide occuring outside the victim's home and with an automatic weapon, the death penalty was actually less likely to be issued if the victim was white. Within this branch, among non-white victims, the death penalty is much more likely to be administered when there are more than 2.5 victims on average. If a handgun was used, and the victim(s) was hispanic, then the death penalty is more likely to be issued if more than 2 people were killed. For non-white victims killed in their homes, the defendant was more likely to be sentenced to death if his education level is unknown.

Unfortunately, the fit of the classification tree is mediocre, so these insights must be taken with a grain of salt. Overall, the algorithm misclassified capital charges at a rate of roughly 44%. The model error for false positives somewhat reflects our cost ratio preferences, but the false positive rate in use is quite disappointing. In addition, a bootstrap standard error estimate indicates that our error rates may vary by roughly 2% over random re-samplings of the data, suggesting that our model may actually be unsuited for drawing serious conclusions.

Summary:

Based on our analysis, it appears that "illegitimate factors" such as race of the defendant, race of the victim, and education level are associated with capital charges with victim counts of 2-3 on average. However, the weak statistical power of this analysis may make it unsuitable for substantive insight. Therefore, in good conscience, we cannot make policy recommendations to the Senate Judiciary Committee without a follow-up study.

Code Appendix:

~~~
---
title: "Death Penalty Analysis"
output: html_notebook
---

Load and prepare dataset
```{r}
load("~/Documents/Coding/Data/MLDeathPenalty.rdata") #load dataset

head(DeathPenalty) #examine dataset
nrow(DeathPenalty) #size of dataset

DeathPenalty$death = as.factor(DeathPenalty$death) #convert numerics to factors
DeathPenalty$gender = as.factor(DeathPenalty$gender)
DeathPenalty$education = as.factor(DeathPenalty$education)
DeathPenalty$birthplace = as.factor(DeathPenalty$birthplace)
DeathPenalty$white = as.factor(DeathPenalty$white)
DeathPenalty$black = as.factor(DeathPenalty$black)
DeathPenalty$hisp = as.factor(DeathPenalty$hisp)
DeathPenalty$working = as.factor(DeathPenalty$working)
DeathPenalty$alcoholhistory = as.factor(DeathPenalty$alcoholhistory)
DeathPenalty$drugshistory = as.factor(DeathPenalty$drugshistory)
DeathPenalty$retarded = as.factor(DeathPenalty$retarded)
DeathPenalty$mentalhistory = as.factor(DeathPenalty$mentalhistory)
DeathPenalty$vmale = as.factor(DeathPenalty$vmale)
DeathPenalty$vwhite = as.factor(DeathPenalty$vwhite)
DeathPenalty$vblack = as.factor(DeathPenalty$vblack)
DeathPenalty$vhisp = as.factor(DeathPenalty$vhisp)
DeathPenalty$bktorture = as.factor(DeathPenalty$bktorture)
DeathPenalty$bkhostage = as.factor(DeathPenalty$bkhostage)
DeathPenalty$bkbeaten = as.factor(DeathPenalty$bkbeaten)
DeathPenalty$bkplead = as.factor(DeathPenalty$bkplead)
DeathPenalty$bksexassault = as.factor(DeathPenalty$bksexassault)
DeathPenalty = DeathPenalty[,-22] #remove duplicate bkplead
DeathPenalty$autogun = as.factor(DeathPenalty$autogun)
DeathPenalty$handgun = as.factor(DeathPenalty$handgun)
DeathPenalty$residence = as.factor(DeathPenalty$residence)
DeathPenalty$vehicle = as.factor(DeathPenalty$vehicle)
DeathPenalty$business = as.factor(DeathPenalty$business)
DeathPenalty$store = as.factor(DeathPenalty$store)
DeathPenalty$stranger = as.factor(DeathPenalty$stranger)
DeathPenalty$rival = as.factor(DeathPenalty$rival)
~~~

~~~
na.indicies = which(is.na(DeathPenalty[,1]), arr.ind=TRUE)
row.na.indicies = unique(na.indicies) #which rows have NA for the response?

DeathPenalty = DeathPenalty[-row.na.indicies,] #remove rows with NAs from the dataframe

anyNA(DeathPenalty[,1]) #check
nrow(DeathPenalty) #new size of dataset
head(DeathPenalty) #examine dataset

```

Univariate statistics
```{r}
plot(DeathPenalty$death, main = "Death Penalty = 1")
plot(DeathPenalty$gender, main = "Gender 1=male 0=female")
plot(DeathPenalty$white, main = "White = 1, other = 0")
plot(DeathPenalty$black, main = "Black = 2, other = 0")
plot(DeathPenalty$hisp, main = "Hispanic = 3, other = 0")
plot(DeathPenalty$education, main = "Edu: hs,ps,college grad =1, nohigh school = 2, unknown
= 3")
plot(DeathPenalty$birthplace, main = "Birthplace US = 1")
plot(DeathPenalty$working, main = "Working = 1, unemp = 0")
plot(DeathPenalty$alcoholhistory, main = "drinking problem = 1")
plot(DeathPenalty$drugshistory, main = "drug history = 1")
plot(DeathPenalty$retarded, main = "retarded = 1")
plot(DeathPenalty$mentalhistory, main = "history mental illness = 1")
plot(DeathPenalty$vmale, main = "victim male = 1")
plot(DeathPenalty$vwhite, main = "victim white = 1")
plot(DeathPenalty$vblack, main = "victim black = 1")
plot(DeathPenalty$vhisp, main = "victim hispanic = 1")
plot(DeathPenalty$bktorture, main = "victim tortured = 1")
plot(DeathPenalty$bkhostage, main = "victim held hostage = 1")
plot(DeathPenalty$bkbeaten, main = "victim beaten = 1")
plot(DeathPenalty$bkplead, main = "victim plead for mercy = 1")
plot(DeathPenalty$bksexassault, main = "victim sexually assaulted = 1")
plot(DeathPenalty$numbervictim, main = "number of homicide victims")
plot(DeathPenalty$autogun, main = "automatic weapon used = 1")
plot(DeathPenalty$handgun, main = "handgun used = 1")
plot(DeathPenalty$residence, main = "victim killed in residence = 1")
plot(DeathPenalty$vehicle, main = "victim killed in vehicle = 1")
plot(DeathPenalty$business, main = "victim killed in their business = 1")
plot(DeathPenalty$store, main = "victim killed in their store = 1 ")
~~~

```
plot(DeathPenalty$stranger, main = "defendant & victim strangers = 1")
plot(DeathPenalty$rival, main= "defendant & victim rivals = 1")

```

Bivariate Statistics

```{r}
plot(DeathPenalty$gender , DeathPenalty$death, xlab = "Gender", ylab = "Death Y/N")
plot(DeathPenalty$white , DeathPenalty$death, xlab = "Race = White", ylab = "Death Y/N")
plot(DeathPenalty$black , DeathPenalty$death, xlab = "Race = Black", ylab = "Death Y/N")
plot(DeathPenalty$hisp , DeathPenalty$death, xlab = "Race = Hispanic", ylab = "Death Y/N")
plot(DeathPenalty$education , DeathPenalty$death, xlab = "Education Level", ylab = "Death
Y/N")
plot(DeathPenalty$birthplace , DeathPenalty$death, xlab = "Birthplace", ylab = "Death Y/N")
plot(DeathPenalty$vmale , DeathPenalty$death, xlab = "Victim Gender", ylab = "Death Y/N")
plot(DeathPenalty$vwhite , DeathPenalty$death, xlab = "Victim White", ylab = "Death Y/N")
plot(DeathPenalty$vblack , DeathPenalty$death, xlab = "Victim Black", ylab = "Death Y/N")
plot(DeathPenalty$vhisp , DeathPenalty$death, xlab = "Victim Hispanic", ylab = "Death Y/N")
```

Split the data into training, evaluation, and test sets:

```{r}

DeathPenalty.split = split(DeathPenalty, rep(1:3, length.out = nrow(DeathPenalty),
                    each = ceiling(nrow(DeathPenalty)/3))) #split the data

DeathPenalty.train = as.data.frame(DeathPenalty.split[1]) #save as new data frames
DeathPenalty.eval = as.data.frame(DeathPenalty.split[2])
DeathPenalty.test = as.data.frame(DeathPenalty.split[3])

colnames(DeathPenalty.train) = colnames(DeathPenalty) #Change column names to match
colnames(DeathPenalty.eval) = colnames(DeathPenalty)
colnames(DeathPenalty.test) = colnames(DeathPenalty)

nrow(DeathPenalty) #check for completeness
nrow(DeathPenalty.train) + nrow(DeathPenalty.eval) + nrow(DeathPenalty.test)


```

Now let's fit many trees to pick the best one

```{r}
library(rpart)
library(rpart.plot)

misclass.rate.store = c() #create vector to store performance metrics

objective.prior.death = sum(DeathPenalty[,1] == 1)/nrow(DeathPenalty) #calculate prior
probabilities
objective.prior.no_death = sum(DeathPenalty[,1] == 0)/nrow(DeathPenalty)
adjusted.prior.death = objective.prior.death * 2
adjusted.prior.no_death = objective.prior.no_death * 1
normalized.prior.death =  adjusted.prior.death/(adjusted.prior.death + adjusted.prior.no_death)
normalized.prior.no_death =  adjusted.prior.no_death/(adjusted.prior.death +
adjusted.prior.no_death)

cp.tune = c(.01,.02,.03,.04,.05) #possible value of the complexity parameter for tuning

for(i in cp.tune){ #fit a tree for each value of cp
  out = rpart(death ~ ., data = DeathPenalty.train, method = "class",
          parms = list(prior = c(normalized.prior.death,normalized.prior.no_death)), cp = i)

  prp(out, extra = 1, faclen = 10, varlen = 15, under = T,
    box.col = c("red","lightblue")[out$frame$yval])

  preds = predict(out, newdata = DeathPenalty.eval, type = "class") #predict values on the
evaluation set

  confusion.matrix = table("Observed Class" = DeathPenalty.eval$death, #confusion matrix
                    "Predicted Class" = preds)[2:1, 2:1]

  print(confusion.matrix)

  misclass.rate = (confusion.matrix[2,1] + confusion.matrix[1,2])/ #misclassification rate
              (confusion.matrix[2,1] + confusion.matrix[1,2] +
                confusion.matrix[1,1] + confusion.matrix[2,2])

  print(misclass.rate)

  misclass.rate.store = append(misclass.rate.store, misclass.rate)
}

which.min(misclass.rate.store) # = 1, so we select the first model to apply to the test data

```

Now let us perform our analysis on the test dataset, cross tabulate the results, and calculate errors:

```{r}
fit = rpart(death ~ ., data = DeathPenalty.train, method = "class",
        parms = list(prior = c(normalized.prior.death, normalized.prior.no_death)), cp = .01)

prp(fit, extra = 1, faclen = 10, varlen = 15, under = T,
   box.col = c("red","lightblue")[out$frame$yval])

fitted.values = predict(out, newdata = DeathPenalty.test, type = "class") #predict values on the
test

confusion.matrix.test = table("Observed Class" = DeathPenalty.test$death, #confusion matrix
                "Predicted Class" = fitted.values)[2:1, 2:1]

print(confusion.matrix.test)

#overall misclassification error
misclass.rate = (confusion.matrix.test[2,1] + confusion.matrix.test[1,2])/ #misclassification rate
           (confusion.matrix.test[2,1] + confusion.matrix.test[1,2] +
             confusion.matrix.test[1,1] + confusion.matrix.test[2,2])

print(misclass.rate)

#use errors
false.positive.use =
confusion.matrix.test[2,1]/(confusion.matrix.test[2,1]+confusion.matrix.test[1,1])
false.negative.use =
confusion.matrix.test[1,2]/(confusion.matrix.test[1,2]+confusion.matrix.test[2,2])
print(false.positive.use)
print(false.negative.use)

#model errors
false.positive.model =
confusion.matrix.test[2,1]/(confusion.matrix.test[2,1]+confusion.matrix.test[2,2])
false.negative.model =
confusion.matrix.test[1,2]/(confusion.matrix.test[1,2]+confusion.matrix.test[1,1])
print(false.positive.model)
print(false.negative.model)

```

Now let's do the bootstrap to get a picture of the uncertainty of our estimates:

```{r}

stdError = function(x){ #define a fn for standard error
  sd(x)/sqrt(length(x))}

ind1 <- sample(nrow(DeathPenalty.test), size=nrow(DeathPenalty.test), replace=TRUE)
x.boot1 <- DeathPenalty.test[ind1,]
head(x.boot1)

fitted.values.1 = predict(out, newdata = x.boot1, type = "class") #predict values on the test

confusion.matrix.1 = table("Observed Class" = x.boot1$death, #confusion matrix
                          "Predicted Class" = fitted.values.1)[2:1, 2:1]

print(confusion.matrix.1)

#overall misclassification error
misclass.rate.1 = (confusion.matrix.1[2,1] + confusion.matrix.1[1,2])/ #misclassification rate
                (confusion.matrix.1[2,1] + confusion.matrix.1[1,2] +
                  confusion.matrix.1[1,1] + confusion.matrix.1[2,2])

print(misclass.rate.1)

#use errors
false.positive.use.1 = confusion.matrix.1[2,1]/(confusion.matrix.1[2,1]+confusion.matrix.1[1,1])
false.negative.use.1 = confusion.matrix.1[1,2]/(confusion.matrix.1[1,2]+confusion.matrix.1[2,2])
print(false.positive.use.1)
print(false.negative.use.1)

#model errors
false.positive.model.1 =
confusion.matrix.1[2,1]/(confusion.matrix.1[2,1]+confusion.matrix.1[2,2])
false.negative.model.1 =
confusion.matrix.1[1,2]/(confusion.matrix.1[1,2]+confusion.matrix.1[1,1])
print(false.positive.model.1)
print(false.negative.model.1)


ind2 <- sample(nrow(DeathPenalty.test), size=nrow(DeathPenalty.test), replace=TRUE)
x.boot2 <- DeathPenalty.test[ind2,]
head(x.boot2)

fitted.values.2 = predict(out, newdata = x.boot2, type = "class") #predict values on the test
```

```
confusion.matrix.2 = table("Observed Class" = x.boot2$death, #confusion matrix
                    "Predicted Class" = fitted.values.2)[2:1, 2:1]

print(confusion.matrix.2)

#overall misclassification error
misclass.rate.2 = (confusion.matrix.2[2,1] + confusion.matrix.2[1,2])/ #misclassification rate
            (confusion.matrix.2[2,1] + confusion.matrix.2[1,2] +
              confusion.matrix.2[1,1] + confusion.matrix.2[2,2])

print(misclass.rate.2)

#use errors
false.positive.use.2 = confusion.matrix.2[2,1]/(confusion.matrix.2[2,1]+confusion.matrix.2[1,1])
false.negative.use.2 = confusion.matrix.2[1,2]/(confusion.matrix.2[1,2]+confusion.matrix.2[2,2])
print(false.positive.use.2)
print(false.negative.use.2)

#model errors
false.positive.model.2 =
confusion.matrix.2[2,1]/(confusion.matrix.2[2,1]+confusion.matrix.2[2,2])
false.negative.model.2 =
confusion.matrix.2[1,2]/(confusion.matrix.2[1,2]+confusion.matrix.2[1,1])
print(false.positive.model.2)
print(false.negative.model.2)

ind3 <- sample(nrow(DeathPenalty.test), size=nrow(DeathPenalty.test), replace=TRUE)
x.boot3 <- DeathPenalty.test[ind3,]
head(x.boot3)

fitted.values.3 = predict(out, newdata = x.boot3, type = "class") #predict values on the test

confusion.matrix.3 = table("Observed Class" = x.boot3$death, #confusion matrix
                    "Predicted Class" = fitted.values.3)[2:1, 2:1]

print(confusion.matrix.3)

#overall misclassification error
misclass.rate.3 = (confusion.matrix.3[2,1] + confusion.matrix.3[1,2])/ #misclassification rate
            (confusion.matrix.3[2,1] + confusion.matrix.3[1,2] +
              confusion.matrix.3[1,1] + confusion.matrix.3[2,2])
```

```
print(misclass.rate.3)

#use errors
false.positive.use.3 = confusion.matrix.3[2,1]/(confusion.matrix.3[2,1]+confusion.matrix.3[1,1])
false.negative.use.3 = confusion.matrix.3[1,2]/(confusion.matrix.3[1,2]+confusion.matrix.3[2,2])
print(false.positive.use.3)
print(false.negative.use.3)

#model errors
false.positive.model.3 =
confusion.matrix.3[2,1]/(confusion.matrix.3[2,1]+confusion.matrix.3[2,2])
false.negative.model.3 =
confusion.matrix.3[1,2]/(confusion.matrix.3[1,2]+confusion.matrix.3[1,1])
print(false.positive.model.3)
print(false.negative.model.3)


ind4 <- sample(nrow(DeathPenalty.test), size=nrow(DeathPenalty.test), replace=TRUE)
x.boot4 <- DeathPenalty.test[ind4,]
head(x.boot4)

fitted.values.4 = predict(out, newdata = x.boot4, type = "class") #predict values on the test

confusion.matrix.4 = table("Observed Class" = x.boot4$death, #confusion matrix
                    "Predicted Class" = fitted.values.4)[2:1, 2:1]

print(confusion.matrix.4)

#overall misclassification error
misclass.rate.4 = (confusion.matrix.4[2,1] + confusion.matrix.4[1,2])/ #misclassification rate
            (confusion.matrix.4[2,1] + confusion.matrix.4[1,2] +
              confusion.matrix.4[1,1] + confusion.matrix.4[2,2])

print(misclass.rate.4)

#use errors
false.positive.use.4 = confusion.matrix.4[2,1]/(confusion.matrix.4[2,1]+confusion.matrix.4[1,1])
false.negative.use.4 = confusion.matrix.4[1,2]/(confusion.matrix.4[1,2]+confusion.matrix.4[2,2])
print(false.positive.use.4)
print(false.negative.use.4)

#model errors
```

```
false.positive.model.4 =
confusion.matrix.4[2,1]/(confusion.matrix.4[2,1]+confusion.matrix.4[2,2])
false.negative.model.4 =
confusion.matrix.4[1,2]/(confusion.matrix.4[1,2]+confusion.matrix.4[1,1])
print(false.positive.model.4)
print(false.negative.model.4)


ind5 <- sample(nrow(DeathPenalty.test), size=nrow(DeathPenalty.test), replace=TRUE)
x.boot5 <- DeathPenalty.test[ind5,]
head(x.boot5)

fitted.values.5 = predict(out, newdata = x.boot5, type = "class") #predict values on the test

confusion.matrix.5 = table("Observed Class" = x.boot5$death, #confusion matrix
                  "Predicted Class" = fitted.values.5)[2:1, 2:1]

print(confusion.matrix.5)

#overall misclassification error
misclass.rate.5 = (confusion.matrix.5[2,1] + confusion.matrix.5[1,2])/ #misclassification rate
           (confusion.matrix.5[2,1] + confusion.matrix.5[1,2] +
             confusion.matrix.5[1,1] + confusion.matrix.5[2,2])

print(misclass.rate.5)

#use errors
false.positive.use.5 = confusion.matrix.5[2,1]/(confusion.matrix.5[2,1]+confusion.matrix.5[1,1])
false.negative.use.5 = confusion.matrix.5[1,2]/(confusion.matrix.5[1,2]+confusion.matrix.5[2,2])
print(false.positive.use.5)
print(false.negative.use.5)

#model errors
false.positive.model.5 =
confusion.matrix.5[2,1]/(confusion.matrix.5[2,1]+confusion.matrix.5[2,2])
false.negative.model.5 =
confusion.matrix.5[1,2]/(confusion.matrix.5[1,2]+confusion.matrix.5[1,1])
print(false.positive.model.5)
print(false.negative.model.5)

stdError(c(misclass.rate.1,misclass.rate.2,misclass.rate.3,misclass.rate.4,misclass.rate.5))
stdError(c(false.positive.use.1, false.positive.use.2, false.positive.use.3, false.positive.use.4,
false.positive.use.5))
```

~~~~
stdError(c(false.negative.use.1,false.negative.use.2,false.negative.use.3,false.negative.use.4,fal
se.negative.use.5))
stdError(c(false.positive.model.1, false.positive.model.2, false.positive.model.3,
false.positive.model.4, false.positive.model.5))
stdError(c(false.negative.model.1, false.negative.model.2, false.negative.model.3,
false.negative.model.4, false.negative.model.5))
```
~~~~