Final Execution Project

Load Packages

```
library(dplyr)
library(readr)
library(broom)
library(ggplot2)
library(stringr)
library(tidymodels)
library(probably)
library(vip)
tidymodels_prefer() # Resolves conflicts, prefers tidymodel functions
```

Data Cleaning

```
executions <- read_csv("executiondata_clean.csv")

executions_clean <- executions %>%
  filter(Victim_Count < 50) %>%
  select(-Name, -County) %>%
  separate(Date, c("Month", "Day", "Year"), sep = "/") %>%
  mutate(VictimMale = case_when(
    str_detect(Victim_Sex, "Male") ~ 1, TRUE ~ 0)) %>% # 1 if there was male victim(s)
  mutate(VictimFemale = case_when(
    str_detect(Victim_Sex, "Female") ~ 1, TRUE ~ 0)) %>% # 1 if there was female victim(s)
  mutate(VictimWhite = case_when(
    str_detect(Victim_Race, 'White') ~ 1, TRUE ~ 0)) %>%
  mutate(VictimLatino = case_when(
    str_detect(Victim_Race, 'Latino') ~ 1, TRUE ~ 0)) %>%
  mutate(VictimBlack = case_when(
    str_detect(Victim_Race, 'Black') ~ 1, TRUE ~ 0)) %>%
  mutate(VictimAsian = case_when(
    str_detect(Victim_Race, 'Asian') ~ 1, TRUE ~ 0))

executions_clean <- executions_clean %>% select(-Victim_Sex, -Day, -Victim_Race)
executions_clean$Month <-
  as.numeric(as.character(executions_clean$Month)) # turn month into numeric variable
executions_clean$Year <-
  as.numeric(as.character(executions_clean$Year)) # turn year into numeric variable

head(executions)
```

```
## # A tibble: 6 x 17
##   Date  Name    Age Sex   Race  Crime Victim_Count Victim_Sex Victim_Race County
##   <chr> <chr> <dbl> <chr> <chr> <chr>        <dbl> <chr>      <chr>       <chr>
## 1 01/1~ Gary~    36 Male  White Murd~            1 Male       White       Utah
## 2 05/2~ John~    30 Male  White Murd~            1 Male       White       Leon
## 3 10/2~ Jess~    46 Male  White Murd~            1 Male       White       Clark
## 4 03/0~ Stev~    24 Male  White Murd~            4 2 Male, 2~ White       Marion
## 5 08/1~ Fran~    38 Male  White Murd~            1 Male       White       Newpo~
## 6 12/0~ Char~    40 Male  Black Murd~            1 Male       White       Tarra~
## # ... with 7 more variables: State <chr>, Region <chr>, Method <chr>,
## #   Juvenile <chr>, Volunteer <chr>, Federal <chr>, Foreign_National <chr>
```

```
head(executions_clean)
```

```
## # A tibble: 6 x 20
##   Month  Year   Age Sex   Race  Crime  Victim_Count State Region Method Juvenile
##   <dbl> <dbl> <dbl> <chr> <chr> <chr>         <dbl> <chr> <chr>  <chr>  <chr>
## 1     1  1977    36 Male  White Murder            1 UT    West   Firin~ No
## 2     5  1979    30 Male  White Murder            1 FL    South  Elect~ No
## 3    10  1979    46 Male  White Murder            1 NV    West   Gas C~ No
## 4     3  1981    24 Male  White Murder            4 IN    Midwe~ Elect~ No
## 5     8  1982    38 Male  White Murder            1 VA    South  Elect~ No
## 6    12  1982    40 Male  Black Murder            1 TX    South  Letha~ No
## # ... with 9 more variables: Volunteer <chr>, Federal <chr>,
## #   Foreign_National <chr>, VictimMale <dbl>, VictimFemale <dbl>,
## #   VictimWhite <dbl>, VictimLatino <dbl>, VictimBlack <dbl>, VictimAsian <dbl>
```

Data Context

The cases represent executed individuals in the United between 1976-2016. The variables used include the Numerical variables: Year of the execution. Age of the executed individual. Victim count of the executed individual.

Categorical variables: Race of executed individual. Sex of the executed individual. Type of crime the executed individual committed The region the execution took place in. The execution method of the executed individual.

The execution database was compiled and published by the Death Penalty Information Center. Victim details, including quantity, sex, and race, were acquired from the Criminal Justice Project's Death Row USA report. The information in this database was obtained from news reports, the Department of Corrections in each state, and the NAACP Legal Defense Fund. We are not able to find context as for why this data was collected or when.

Research Questions

Regression

Can we accurately predict the number of victims that an executed individual had based on their execution data?

Our motivation is seeing how execution and demographic information might correlate with the severity of the crime (in terms of people murdered).

Classification

We are looking to build a classification model that will predict the method of execution from our data set. To do this, we will use a logistic regression model and a random forests model. Our main aim is to determine if there is a specific method of execution that is more common for certain variables.

In our logistic regression model we focus on if an individual was electrocuted or not since it is arguably one of the most painful and long-lasting methods of execution.

In our random forest model, we attempt to predict between lethal injection, hanging, firing squad, and electrocution.

Unsupervised Learning

What do different executions have in common based on Victim Count and Year executed?

Regression

Methods

Models Used

In order to approach the question of how execution methods and demographics is related to the number of victims an executed person has, we decided to use ordinary least squares regression and LASSO regression. The ordinary least squares model allows us to use every variable in the model and directly comprehend which variables have positive or negative correlations with number of victims and by how much. LASSO was used as a secondary method in order to create a model that was less likely to overfit to data by eliminating variables and decreasing variance. This also allows us to find the factors most important to the variation in victim counts based on those variables which still exist after LASSO tuning.

Model Evaluation

In order to evaluate the OLS model, cross validation was used to train and test the model and create summary statistics on error such as root mean squared error, mean average error, etc. We also created residual plots of numerical variables versus residuals to find out if non-linear transformations were necessary and residual versus predicted plots to see trends in over or under predicting.

For the LASSO model, we used the same evaluation metrics as above but also used a tuning plot on root mean squared error in order to calculate the best penalty value by taking the value with the least RMSE.

Results

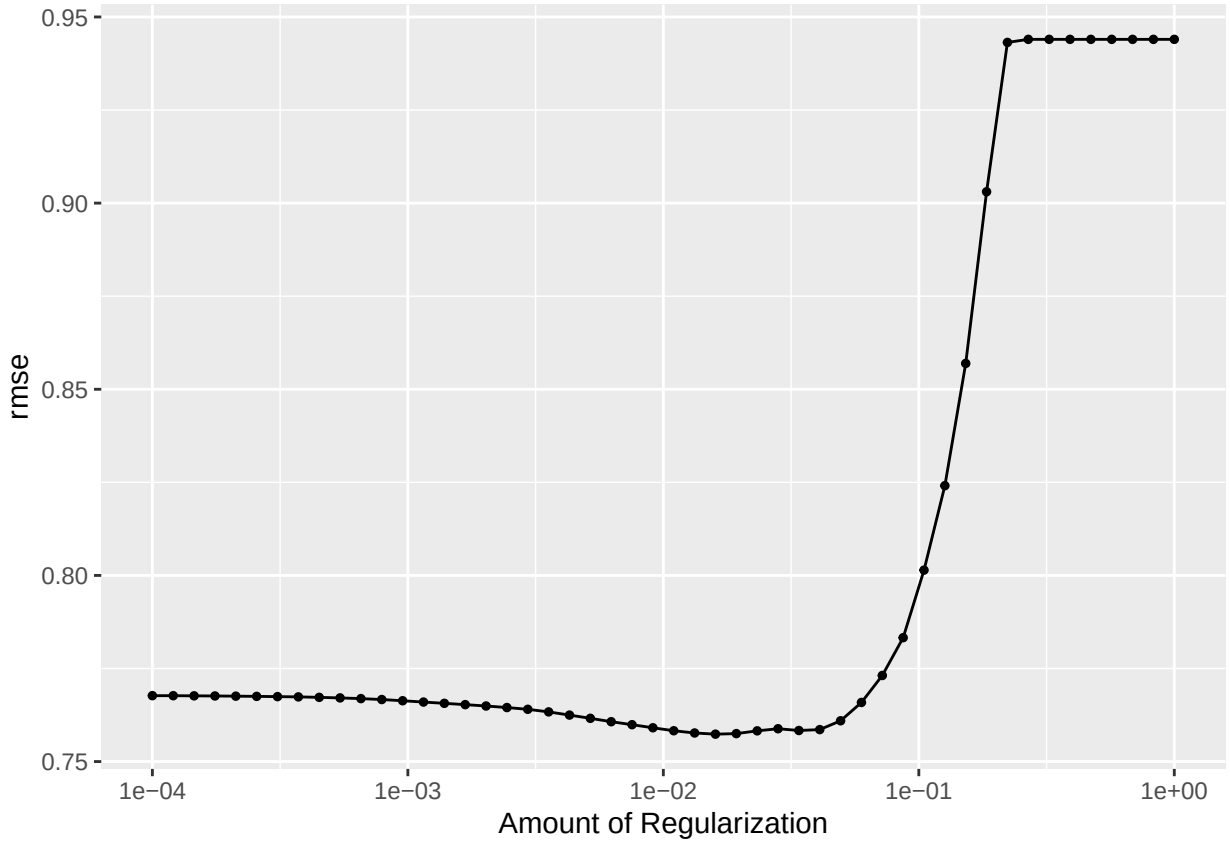
Model Creation

```
set.seed(123)
execute_cv <- vfold_cv(executions_clean, v=10)
```

```r
# OLS spec
lm_spec <- linear_reg() %>%
  set_engine(engine = 'lm') %>%
    set_mode('regression')

# LASSO spec
lm_lasso_spec <-
  linear_reg() %>%
  set_args(mixture = 1, penalty = tune()) %>%
  set_engine(engine = 'glmnet') %>%
  set_mode('regression')
```

```r
# create recipe
execute_rec <- recipe(Victim_Count~., data = executions_clean) %>%
  step_cut(Month, breaks=c(1, 3, 8, 12)) %>% # break months into winter, spring, summer, fall
  step_nzv(all_numeric_predictors()) %>% # remove near zero variance values
  step_corr(all_numeric_predictors()) %>% # remove correlated numeric variables
  step_normalize(all_numeric_predictors()) %>% # normalize numeric variables
  step_unknown(all_nominal_predictors()) %>% # mark NA for unknown nominal variables
  step_dummy(all_nominal_predictors()) # create indicator variables for all nominal variables

# create workflows
execute_wf <- workflow() %>%
  add_recipe(execute_rec) %>%
  add_model(lm_spec)

lasso_wf <- workflow() %>%
  add_recipe(execute_rec) %>%
  add_model(lm_lasso_spec)
```

```r
# tune LASSO penalty term
penalty_grid <- grid_regular(
  penalty(range = c(-4, 0)), # range picked based on autoplot trial and error
  levels = 50
)

tune_res <- tune_grid(
  lasso_wf,
  resamples = execute_cv,
  metrics = metric_set(rmse),
  grid = penalty_grid
)

autoplot(tune_res) # to find a good range of values to tune from, can be commented out
```




```
collect_metrics(tune_res) %>%
  filter(.metric == 'rmse') %>% # using root mean squared error
  select(penalty, rmse = mean)
```

```
## # A tibble: 50 x 2
##     penalty  rmse
##       <dbl> <dbl>
##  1 0.0001   0.768
##  2 0.000121 0.768
##  3 0.000146 0.768
##  4 0.000176 0.768
##  5 0.000212 0.768
##  6 0.000256 0.768
##  7 0.000309 0.767
##  8 0.000373 0.767
##  9 0.000450 0.767
## 10 0.000543 0.767
## # ... with 40 more rows
```

```
best_penalty <- select_best(tune_res, metric = 'rmse')

lasso_tuned_wf <- finalize_workflow(lasso_wf, best_penalty)
```

```
# fit models
fit_model <- fit(execute_wf, data = executions_clean)
tidy(fit_model)
```

```
## # A tibble: 72 x 5
##    term          estimate std.error statistic   p.value
##    <chr>            <dbl>     <dbl>     <dbl>     <dbl>
##  1 (Intercept)     1.36      0.413      3.29  1.03e-  3
##  2 Year            0.0108    0.0261     0.412 6.80e-  1
##  3 Age             0.0291    0.0236     1.23  2.18e-  1
##  4 VictimMale      0.757     0.0312    24.2   1.55e-108
##  5 VictimFemale    0.742     0.0312    23.8   5.62e-105
##  6 VictimWhite     0.239     0.0435     5.50  4.47e-  8
##  7 VictimLatino    0.165     0.0325     5.07  4.51e-  7
##  8 VictimBlack     0.270     0.0394     6.85  1.07e- 11
##  9 Month_X.3.8.    0.0648    0.0508     1.28  2.02e-  1
## 10 Month_X.8.12.   0.0199    0.0562     0.354 7.23e-  1
## # ... with 62 more rows
```

```
fit_model_lasso <- fit(lasso_tuned_wf, data = executions_clean)
tidy(fit_model_lasso)
```

```
## # A tibble: 72 x 3
##    term          estimate penalty
##    <chr>            <dbl>   <dbl>
##  1 (Intercept)     1.31    0.0160
##  2 Year            0       0.0160
##  3 Age             0.0196  0.0160
##  4 VictimMale      0.715   0.0160
##  5 VictimFemale    0.701   0.0160
##  6 VictimWhite     0.104   0.0160
##  7 VictimLatino    0.0768  0.0160
##  8 VictimBlack     0.144   0.0160
##  9 Month_X.3.8.    0.0314  0.0160
## 10 Month_X.8.12.   0       0.0160
## # ... with 62 more rows
```

Evaluation

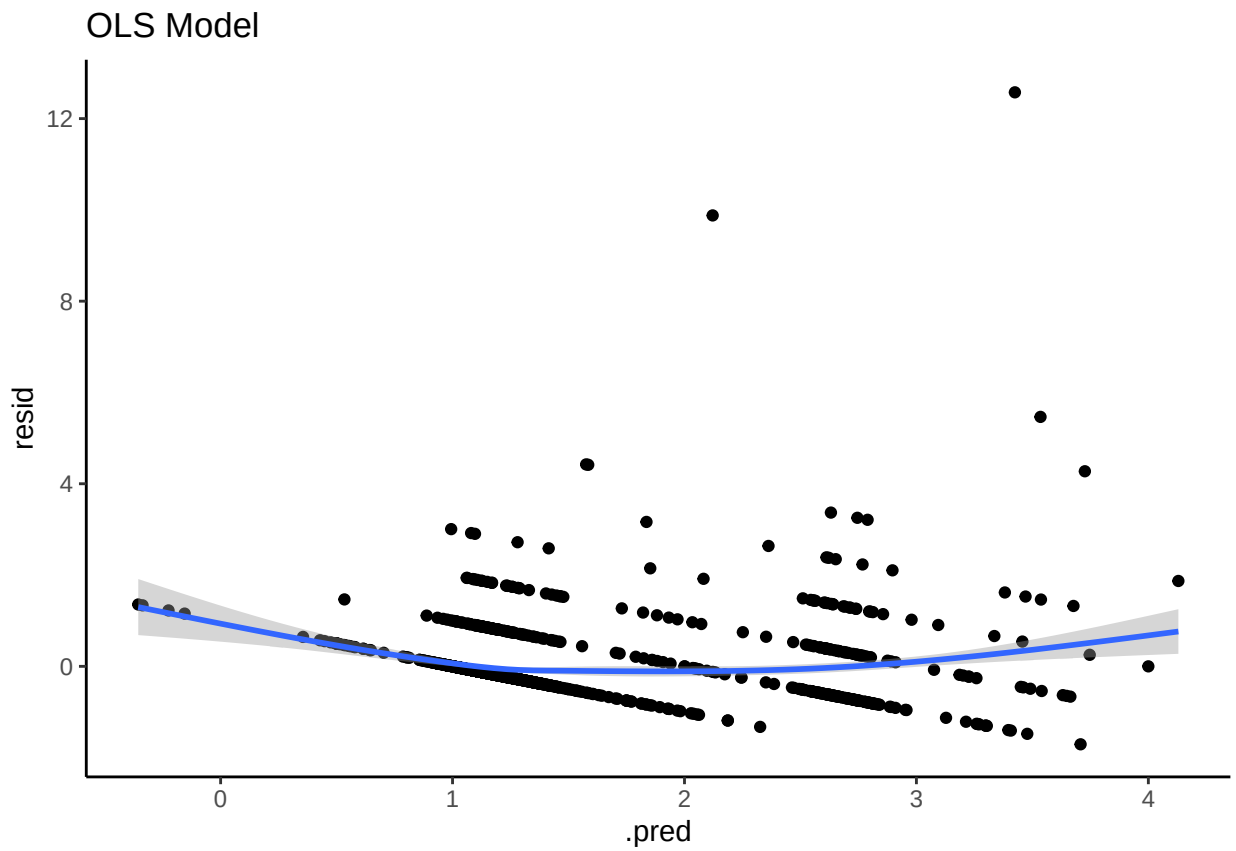
```
# fit models using cross validation
fit_resamples(execute_wf,
      resamples = execute_cv,
      metrics = metric_set(rmse, rsq, mae))  %>%
      collect_metrics(summarize = TRUE)
```

```
## # A tibble: 3 x 6
##   .metric .estimator  mean     n std_err .config
##   <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1 mae     standard   0.418    10  0.0233 Preprocessor1_Model1
## 2 rmse    standard   0.768    10  0.0846 Preprocessor1_Model1
## 3 rsq     standard   0.378    10  0.0300 Preprocessor1_Model1
```

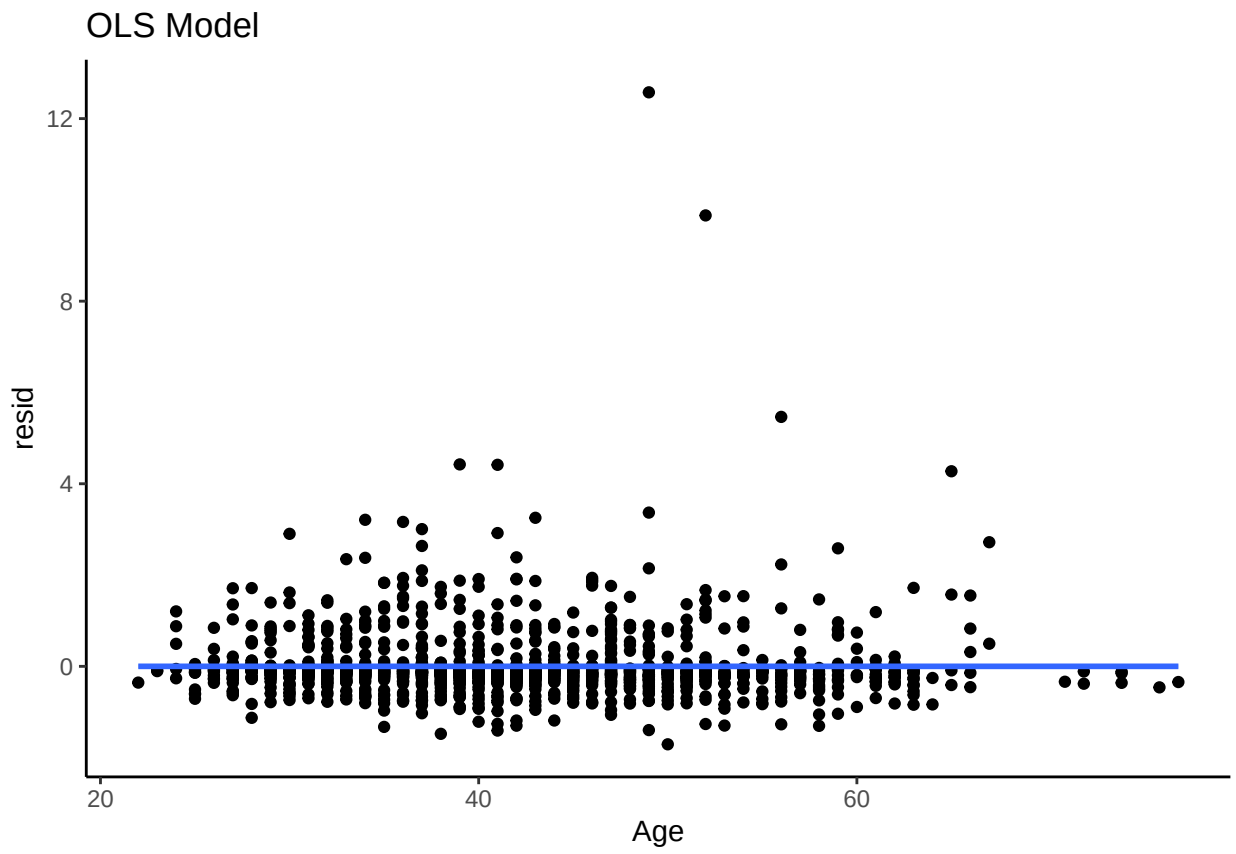
```
fit_resamples(lasso_tuned_wf,
      resamples = execute_cv,
      metrics = metric_set(rmse, rsq, mae))  %>%
      collect_metrics(summarize = TRUE)
```

```
## # A tibble: 3 x 6
##   .metric .estimator  mean     n std_err .config
##   <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1 mae     standard   0.416    10  0.0226 Preprocessor1_Model1
## 2 rmse    standard   0.757    10  0.0903 Preprocessor1_Model1
## 3 rsq     standard   0.385    10  0.0320 Preprocessor1_Model1
```

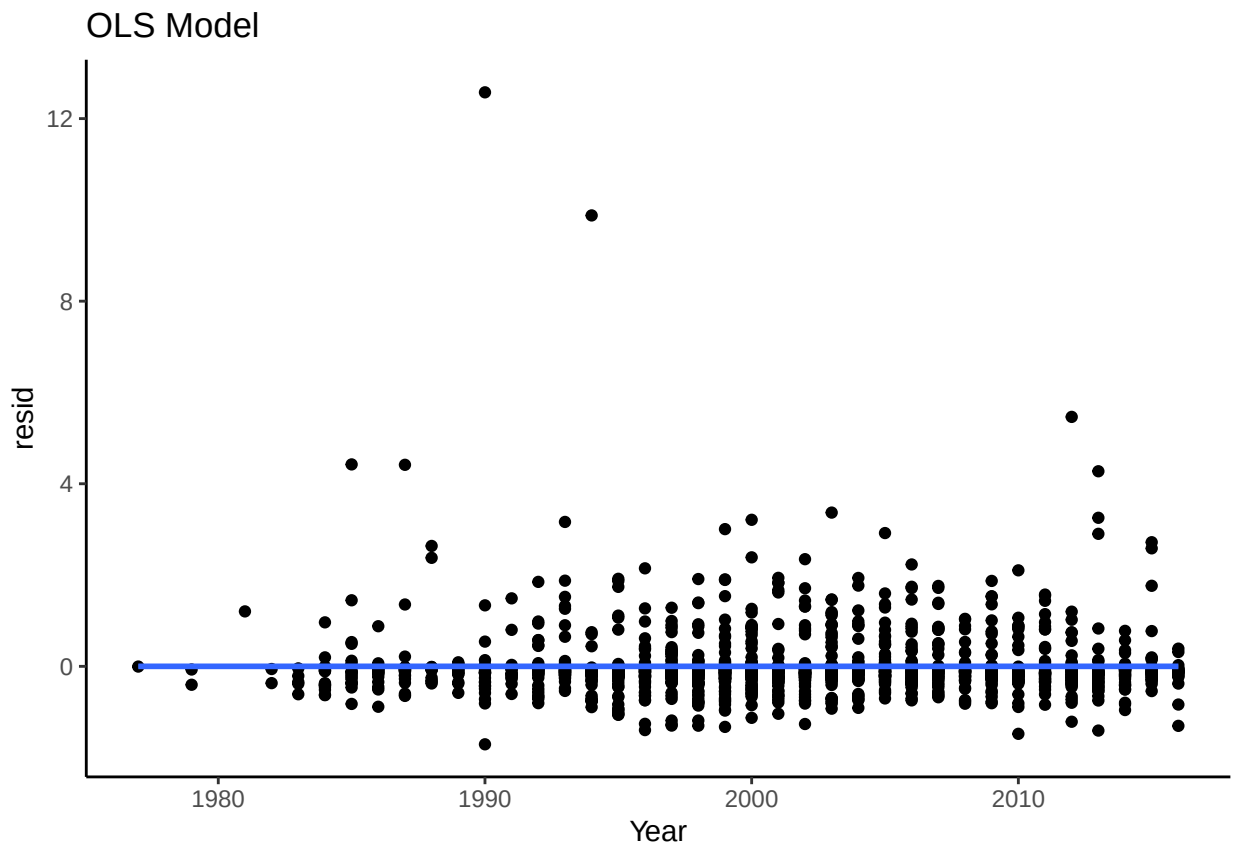
```
# OLS MODEL
# create data frame with predicted values and residuals
mod_output <- fit_model %>%
    predict(new_data = executions_clean) %>%
    bind_cols(executions_clean) %>%
    mutate(resid = Victim_Count - .pred)

# residual versus predicted scatter plot
mod_output %>%
  ggplot(aes(x = .pred, y = resid)) +
    geom_point() +
    geom_smooth()+
    ggtitle("OLS Model") +
    theme_classic()
```



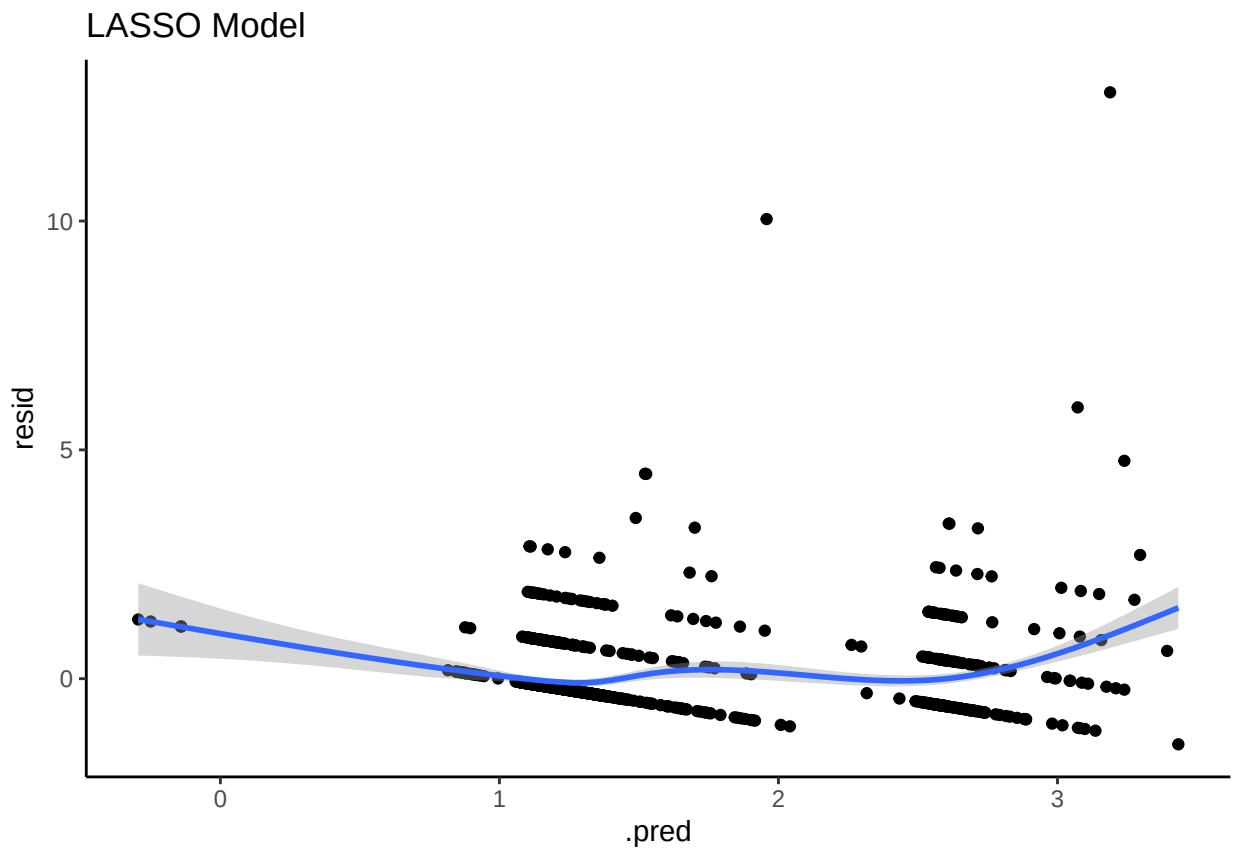

```
# residual versus age scatter plot
mod_output %>%
  ggplot(aes(x = Age, y = resid)) +
    geom_point() +
    geom_smooth() +
    ggtitle("OLS Model") +
    theme_classic()
```



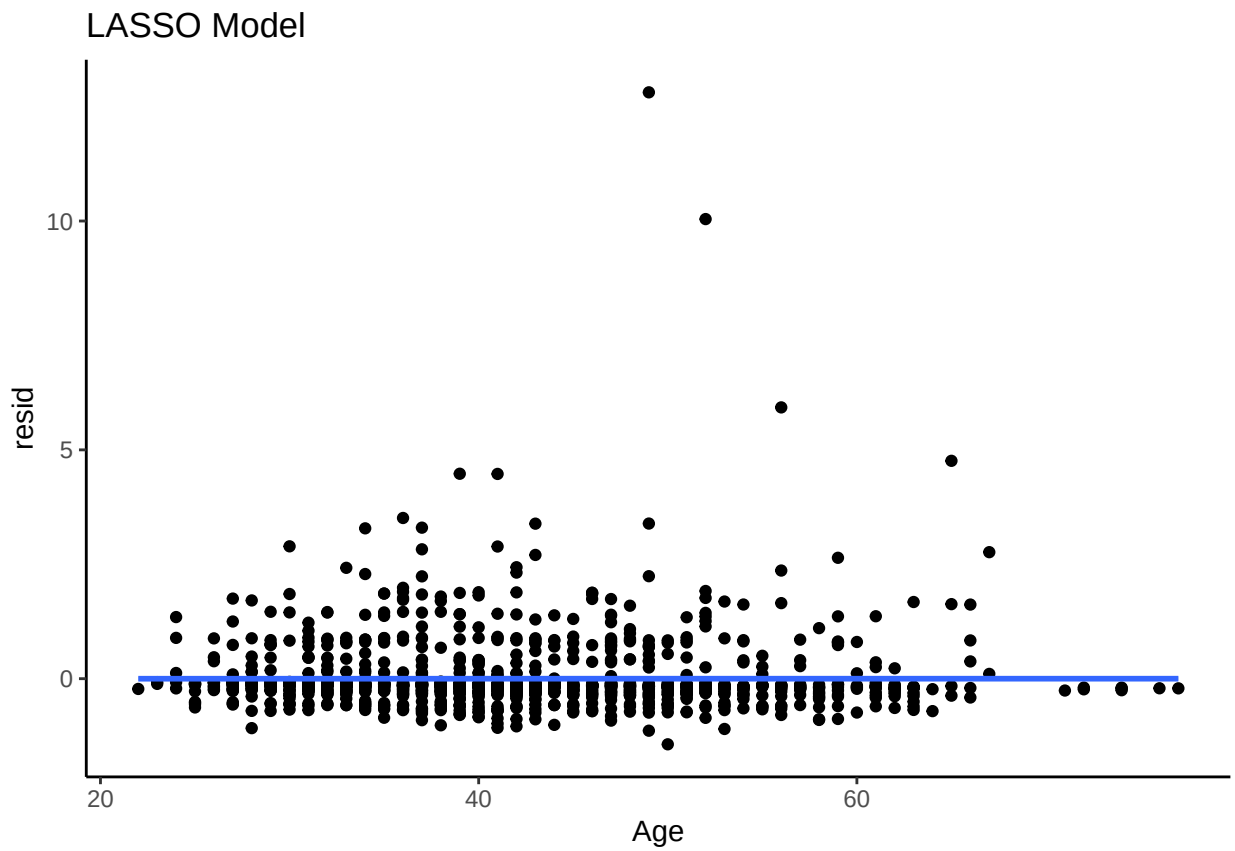

```
# residual versus year scatter plot
mod_output %>%
  ggplot(aes(x = Year, y = resid)) +
    geom_point() +
    geom_smooth() +
    ggtitle("OLS Model") +
    theme_classic()
```



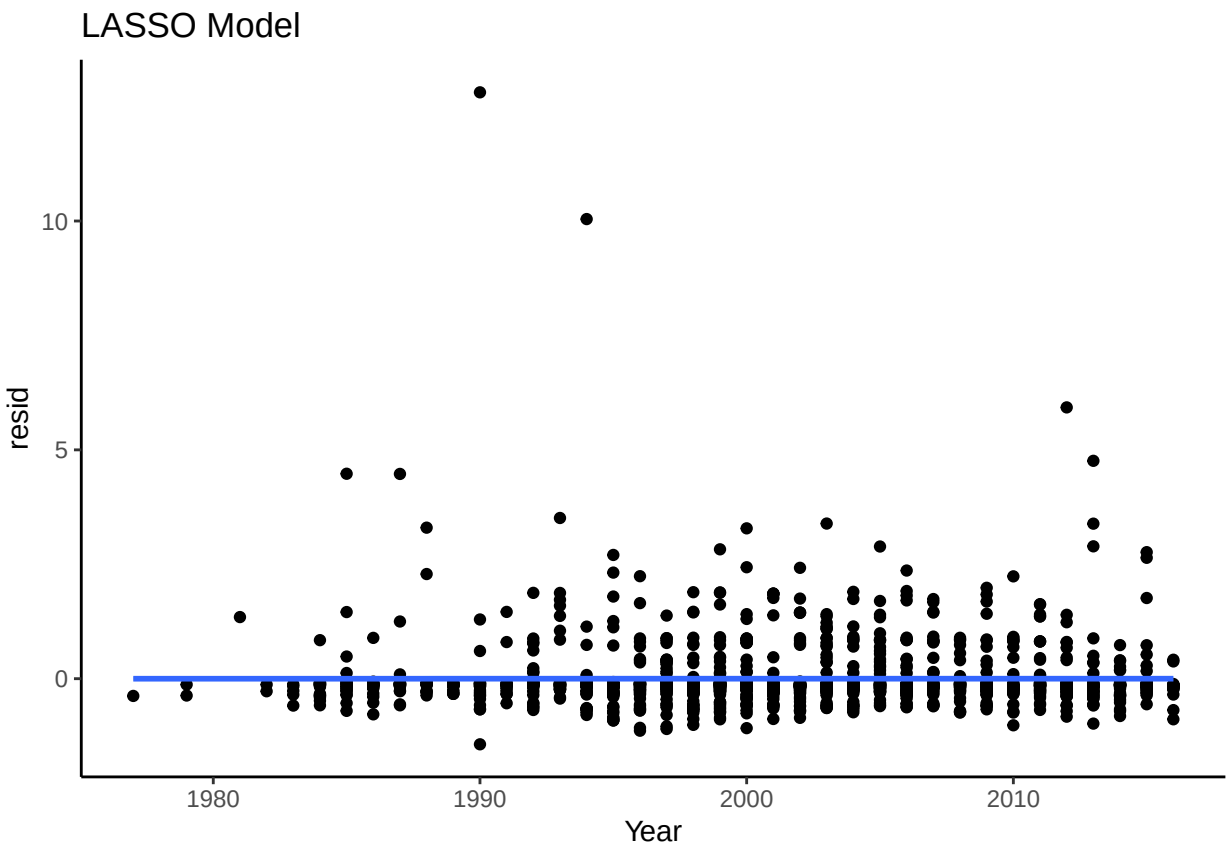

```
## LASSO MODEL
# create data frame with predicted values and residuals
mod_output_2 <- fit_model_lasso %>%
    predict(new_data = executions_clean) %>%
    bind_cols(executions_clean) %>%
    mutate(resid = Victim_Count - .pred)

# residual versus predicted scatter plot
mod_output_2 %>%
  ggplot(aes(x = .pred, y = resid)) +
    geom_point() +
    geom_smooth() +
    ggtitle("LASSO Model") +
    theme_classic()
```




```
# residual versus age scatter plot
mod_output_2 %>%
  ggplot(aes(x = Age, y = resid)) +
    geom_point() +
    geom_smooth() +
    ggtitle("LASSO Model") +
    theme_classic()
```




```
# residual versus year scatter plot
mod_output_2 %>%
  ggplot(aes(x = Year, y = resid)) +
    geom_point() +
    geom_smooth() +
    ggtitle("LASSO Model") +
    theme_classic()
```



Conclusion

Final Model: LASSO Model

```
fit_resamples(lasso_tuned_wf,
      resamples = execute_cv,
      metrics = metric_set(rmse, rsq, mae))  %>%
      collect_metrics(summarize = TRUE)
```

```
## # A tibble: 3 x 6
##   .metric .estimator  mean     n std_err .config
##   <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1 mae     standard   0.416    10  0.0226 Preprocessor1_Model1
## 2 rmse    standard   0.757    10  0.0903 Preprocessor1_Model1
## 3 rsq     standard   0.385    10  0.0320 Preprocessor1_Model1
```

Since the cross validated error metrics above were almost the same in the LASSO model as the OLS model, we chose to use LASSO since it uses less variables and is less prone to excessive variance and overfitting.

The mean absolute error for the tuned LASSO model has a mean of 0.42, meaning that on average, the LASSO model predicts incorrectly by about 0.42 victims. The standard error for this value is low at 0.02, meaning it is likely that the LASSO model is predicting somewhere between 0.40 and 0.44 victims of the actual amount that an executed individual has. We have determined that this is an acceptable amount of error, since we measure victims in whole units, and it is unlikely to be off by more than 1 victim.

```r
# Obtain the predictors and coefficients of the "best" model
# Filter out the coefficient are 0
final_fit <- fit_model_lasso %>% tidy() %>% filter(estimate != 0)

final_fit
```

```
## # A tibble: 37 x 3
##    term          estimate penalty
##    <chr>            <dbl>   <dbl>
##  1 (Intercept)   1.31      0.0160
##  2 Age           0.0196    0.0160
##  3 VictimMale    0.715     0.0160
##  4 VictimFemale  0.701     0.0160
##  5 VictimWhite   0.104     0.0160
##  6 VictimLatino  0.0768    0.0160
##  7 VictimBlack   0.144     0.0160
##  8 Month_X.3.8.  0.0314    0.0160
##  9 Sex_Male      0.0651    0.0160
## 10 Race_Black   -0.000722  0.0160
## # ... with 27 more rows
```

The variables for executions in Illinois and Conneticut were correlated to the largest increase in predicted victim count, holding other variables constant. On the other hand, the execution being in the South or being done by firing squad had the most negative correlation with number of victims. One of the more interesting results was in comparing coefficients of executed individual race and victim race. An executed individual who was black was correlated to a small decrease in victim count holding other variables constant, whereas having a black victim correlated to having more victims holding other variables constant. On the other hand, an executed individual being white correlated to a slight predicted increase in victim count holding other variables constant, along with if the victim(s) were white. Other demographic coefficients also paint stories of how execution is handled in the US.

```r
glmnet_output <- fit_model_lasso %>% extract_fit_engine()

# Create a boolean matrix (predictors x lambdas) of variable exclusion
bool_predictor_exclude <- glmnet_output$beta==0

# Loop over each variable
var_imp <- sapply(seq_len(nrow(bool_predictor_exclude)), function(row) {
    this_coeff_path <- bool_predictor_exclude[row,]
    if(sum(this_coeff_path) == ncol(bool_predictor_exclude)){ return(0)}else{
    return(ncol(bool_predictor_exclude) - which.min(this_coeff_path) + 1)}
})

# Create a dataset of this information and sort
var_imp_data <- tibble(
    var_name = rownames(bool_predictor_exclude),
    var_imp = var_imp
)
var_imp_data %>% arrange(desc(var_imp)) # most important variable first
```

```
## # A tibble: 71 x 2
##    var_name         var_imp
```

```
##    <chr>              <dbl>
##  1 VictimMale            82
##  2 VictimFemale          82
##  3 State_AR              72
##  4 State_IL              72
##  5 VictimBlack           71
##  6 State_CA              71
##  7 State_CT              71
##  8 State_FL              69
##  9 Volunteer_Yes         69
## 10 Region_Northeast      68
## # ... with 61 more rows
```

Having a male victim or female victim are the two most persistent variables in the LASSO model, showing that sex of victims are one of the most important indicators of victim counts. After that come Arkansas and Illinois execution which is slightly less of interest to us in our research as it likely points to where criminal activity and/or executions often take place. The variable tied for 3rd most important variable, if the executed individual had a black victim, is much more interesting. The correlation between having a black victim and having more victims is important to this model.

Classification

Methods

In our logistic regression model we focus on if an individual was electrocuted or not since it is arguably one of the most painful and long-lasting methods of execution.

In our random forest model, we attempt to predict between lethal injection, hanging, firing squad, and electrocution.

In order to evaluate the logistic model we calculated accuracy, specificity, and sensitivity along with ROC Area Under the Curve. For the random forest model we looked at accuracy and out of bag error in order to evaluate the model.

These two models give us insight on how to predict execution types based on execution demographics. What leads to execution by electrocution? Who is likely to be executed by lethal injection? These models aim to show the most important variables in these decisions.

Results

```
executions_clean <- executions %>%
  select(-Name, -County) %>%
  filter(Victim_Count < 50) %>%
  separate(Date, c("Month", "Day", "Year"), sep = "/") %>%
  mutate(VictimMale = case_when(
    str_detect(Victim_Sex, "Male") ~ 1, TRUE ~ 0)) %>% # 1 if there was male victim(s)
  mutate(VictimFemale = case_when(
    str_detect(Victim_Sex, "Female") ~ 1, TRUE ~ 0)) %>% # 1 if there was female victim(s)
  mutate(VictimWhite = case_when(
    str_detect(Victim_Race, 'White') ~ 1, TRUE ~ 0)) %>%
  mutate(VictimLatino = case_when(
```

```
    str_detect(Victim_Race, 'Latino') ~ 1, TRUE ~ 0)) %>%
  mutate(VictimBlack = case_when(
    str_detect(Victim_Race, 'Black') ~ 1, TRUE ~ 0)) %>%
  mutate(VictimAsian = case_when(
    str_detect(Victim_Race, 'Asian') ~ 1, TRUE ~ 0)) %>%
  mutate(Electrocution = case_when(
    str_detect(Method, 'Electrocution') ~ 'yes', TRUE ~ 'no')) # make a variable that shows if the pers


executions_clean <- executions_clean %>% select(-Victim_Sex, -Day, -Victim_Race)
executions_clean$Month <-
  as.numeric(as.character(executions_clean$Month)) # turn month into numeric variable
executions_clean$Year <-
  as.numeric(as.character(executions_clean$Year)) # turn year into numeric variable
executions_clean$Electrocution <- factor(executions_clean$Electrocution, ordered = FALSE )
executions_clean <- executions_clean %>%
  mutate(Electrocution = relevel(Electrocution, ref='no')) #set reference level

head(executions)
```

```
## # A tibble: 6 x 17
##   Date  Name    Age Sex   Race  Crime Victim_Count Victim_Sex Victim_Race County
##   <chr> <chr> <dbl> <chr> <chr> <chr>        <dbl> <chr>      <chr>       <chr>
## 1 01/1~ Gary~    36 Male  White Murd~            1 Male       White       Utah
## 2 05/2~ John~    30 Male  White Murd~            1 Male       White       Leon
## 3 10/2~ Jess~    46 Male  White Murd~            1 Male       White       Clark
## 4 03/0~ Stev~    24 Male  White Murd~            4 2 Male, 2~ White       Marion
## 5 08/1~ Fran~    38 Male  White Murd~            1 Male       White       Newpo~
## 6 12/0~ Char~    40 Male  Black Murd~            1 Male       White       Tarra~
## # ... with 7 more variables: State <chr>, Region <chr>, Method <chr>,
## #   Juvenile <chr>, Volunteer <chr>, Federal <chr>, Foreign_National <chr>
```

```
head(executions_clean)
```

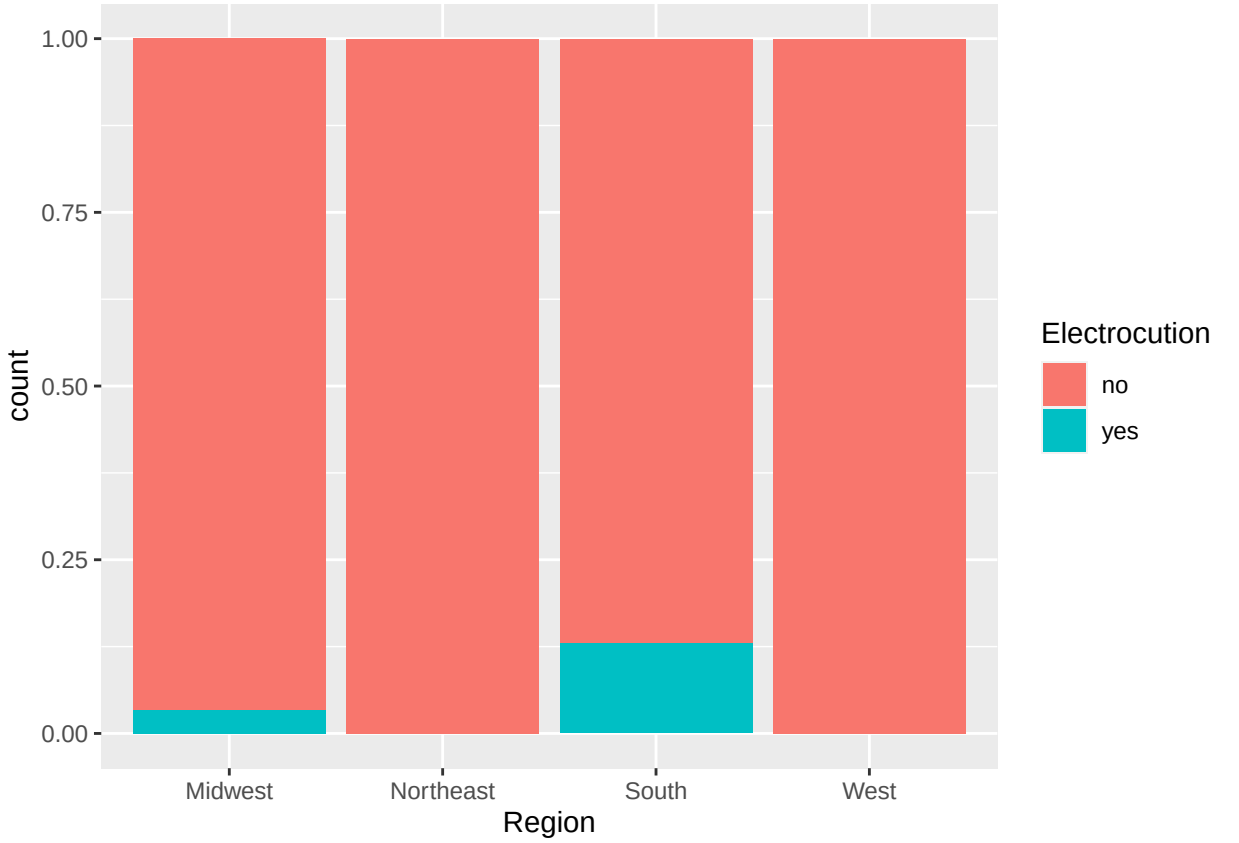
```
## # A tibble: 6 x 21
##   Month  Year   Age Sex   Race  Crime  Victim_Count State Region Method Juvenile
##   <dbl> <dbl> <dbl> <chr> <chr> <chr>         <dbl> <chr> <chr>  <chr>  <chr>
## 1     1  1977    36 Male  White Murder            1 UT    West   Firin~ No
## 2     5  1979    30 Male  White Murder            1 FL    South  Elect~ No
## 3    10  1979    46 Male  White Murder            1 NV    West   Gas C~ No
## 4     3  1981    24 Male  White Murder            4 IN    Midwe~ Elect~ No
## 5     8  1982    38 Male  White Murder            1 VA    South  Elect~ No
## 6    12  1982    40 Male  Black Murder            1 TX    South  Letha~ No
## # ... with 10 more variables: Volunteer <chr>, Federal <chr>,
## #   Foreign_National <chr>, VictimMale <dbl>, VictimFemale <dbl>,
## #   VictimWhite <dbl>, VictimLatino <dbl>, VictimBlack <dbl>,
## #   VictimAsian <dbl>, Electrocution <fct>
```

Logistic Regression

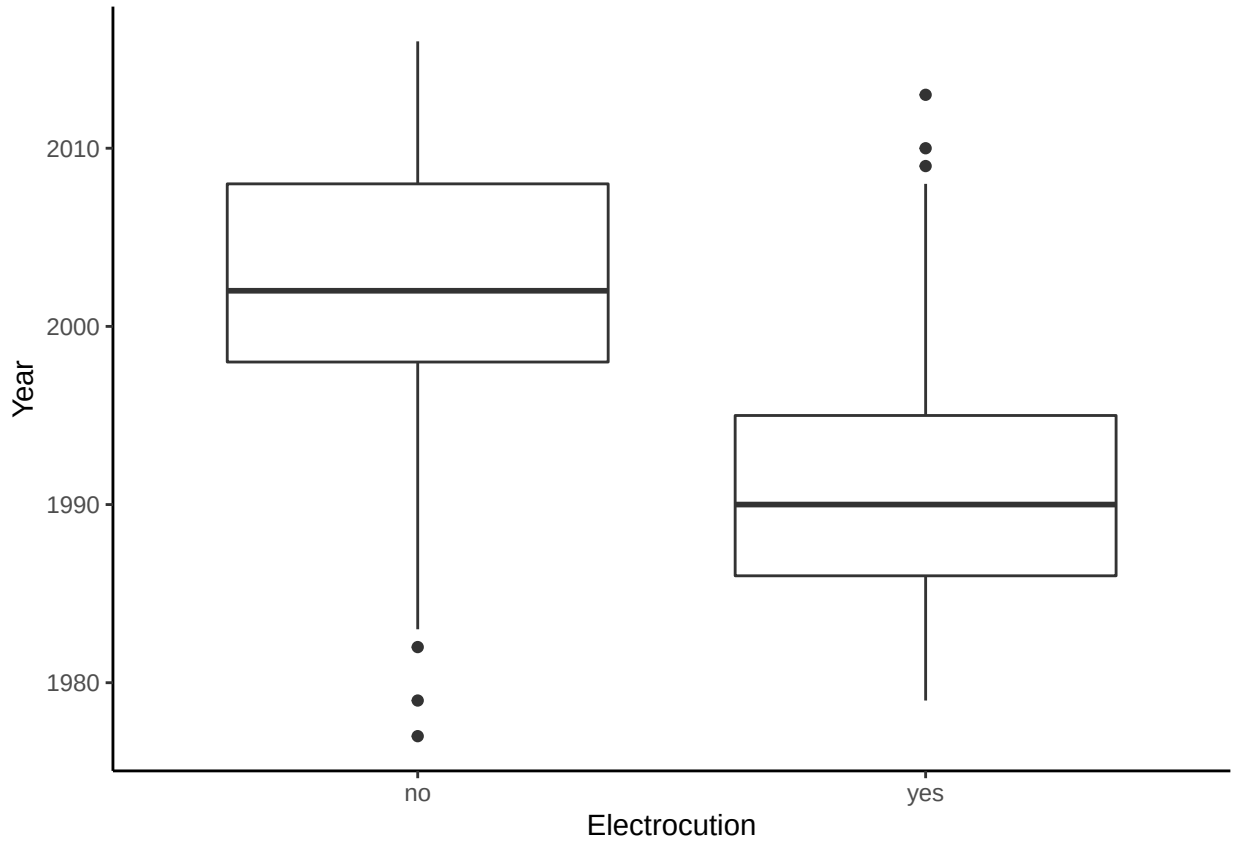
Visualizations of Predictive Ability We determined the variables with the most predictive ability are region, year, and race. The first graph shows that electrocution has only been used in the Midwest and South,

with the South having a much higher use. The second graph shows that electrocution has mainly been used as a method of execution prior to 2000. Graph 3 shows that electrocution has been used as a method of execution primarily for Black, Native American, and White individuals and for a small percentage of Latino individuals. Black individuals are the most common racial group to be executed with electrocution. The fourth and final graph shows the correlation between region and race. The Midwest and the South are the only two regions to have used electrocution as a method of execution, and have executed the vast majority of individuals in the data set. Most importantly, these two regions have executed the majority of Black individuals.

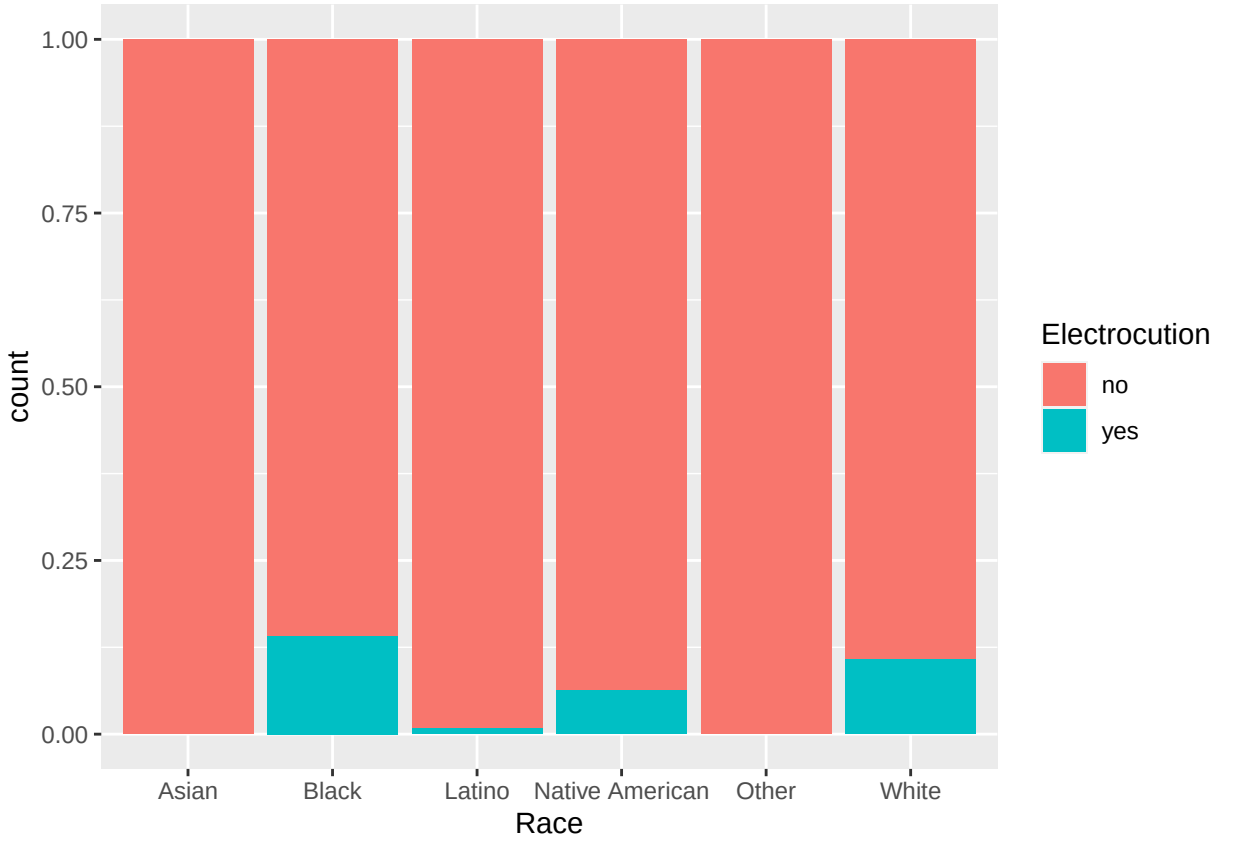
```
# bar chart of region versus electrocution
ggplot(executions_clean, aes(x = Region, fill = Electrocution)) +
    geom_bar(position = 'fill')
```



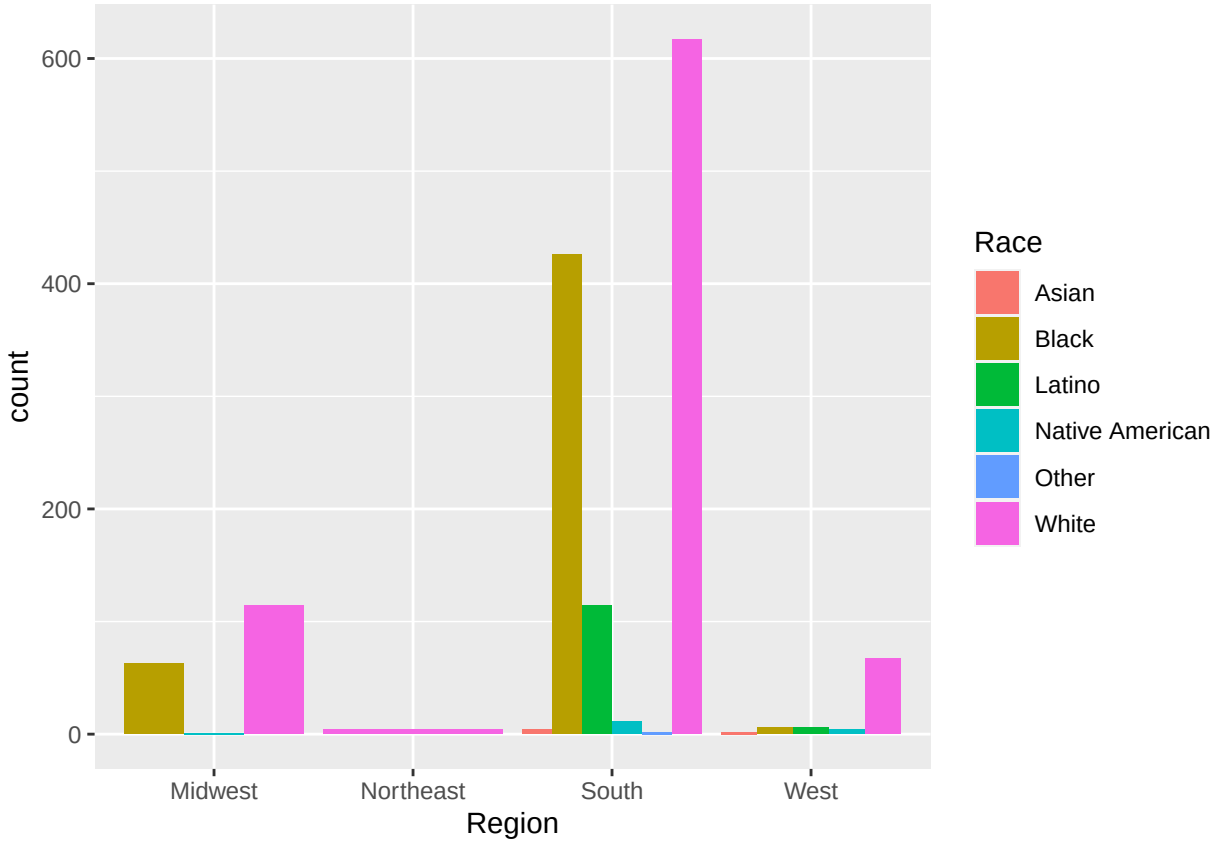

```
# box plot of year versus electrocution
ggplot(executions_clean, aes(x = Electrocution, y = Year)) +
  geom_boxplot() +
  theme_classic()
```




```
# bar chart of race versus electrocution
ggplot(executions_clean, aes(x = Race, fill = Electrocution)) +
  geom_bar(position = 'fill')
```




```
# bar chart of race versus region
ggplot(executions_clean, aes(x = Region, fill = Race)) +
  geom_bar(position = 'dodge')
```

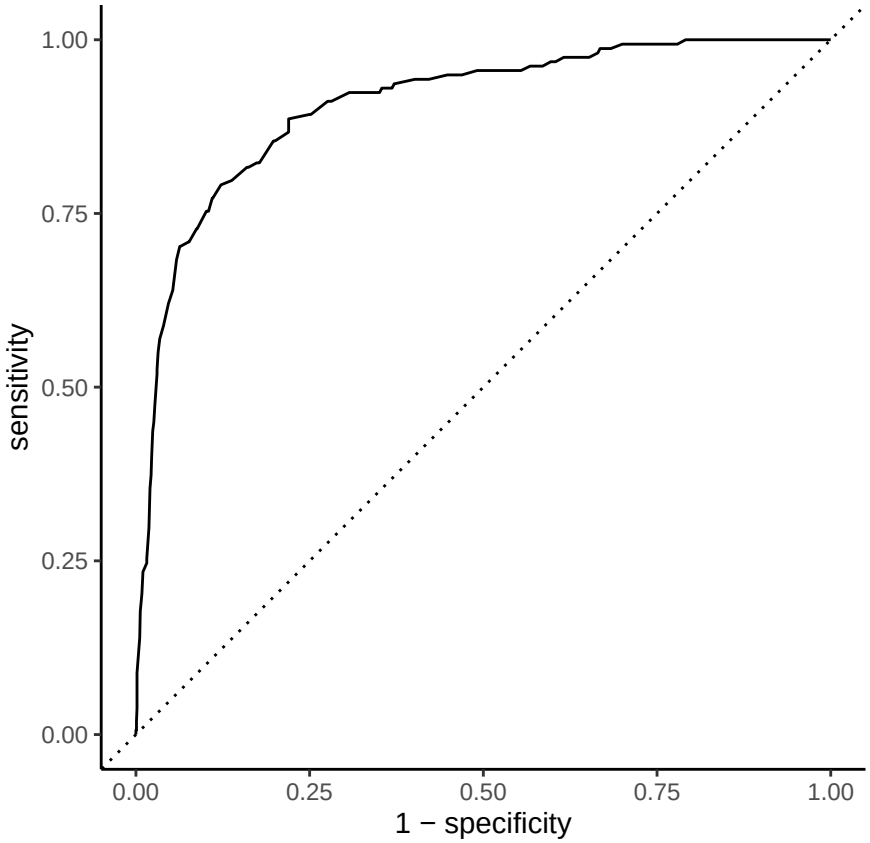



```
# create specification for logistic model

logistic_spec <- logistic_reg() %>%
    set_engine('glm') %>%
    set_mode('classification')

#recipe using important variables
logistic_rec <- recipe(Electrocution ~ Year + Region + Race, data = executions_clean)

#create a workflow
log_wf <- workflow() %>%
    add_recipe(logistic_rec) %>%
    add_model(logistic_spec)
#fit the model to all data
log_fit <- fit(log_wf, data = executions_clean)
```

Implementing Logistic Regression

Evaluating the Model This graph of the ROC curve shows that the model is accurate, but not perfect.

```r
# create prediction table
logistic_output <-  executions_clean %>%
  bind_cols(predict(log_fit, new_data = executions_clean, type = 'prob'))

logistic_roc <- logistic_output %>%
    roc_curve(Electrocution, .pred_yes, event_level = "second")
# plot ROC AUC
autoplot(logistic_roc) + theme_classic()
```



The model is overall 92% accurate. However, it is important to note that just under 11% of the executions in the data set used electrocution, creating a NIR of 89%. Furthermore, the model has a sensitivity of 57% which shows that there are nearly the same number of true positives as false negatives. The model is under-predicting electrocution as a method of execution. The model has a much higher specificity at 96.5%. This is due to the high number of true negatives in the data set.

```r
logistic_output <-  executions_clean %>%
  bind_cols(predict(log_fit, new_data = executions_clean, type = 'prob'))

# using threshold of 0.63 make hard predictions
logistic_output <- logistic_output %>%
  mutate(.pred_class = make_two_class_pred(`.pred_no`, levels(Electrocution), threshold = 0.63))

# output truth v prediction table
logistic_output %>%
  conf_mat(truth = Electrocution, estimate = .pred_class)
```

```
##            Truth
## Prediction   no  yes
##        no  1239   68
##        yes   44   90
```

Looking at cross validated test metrics, the model's accuracy and specificity remain almost the same. Interestingly the sensitivity drops to 46%, meaning the cross validated data shows an even higher rate of predicting false negatives. It is also important to note that the AUC value is 0.900, which is greater than expected for such a low sensitivity rate.

```
set.seed(123)
data_cv10 <- vfold_cv(executions_clean, v = 10)

# fit to 10 fold cv samples
log_cv_fit <- fit_resamples(
    log_wf,
    resamples = data_cv10,
    metrics = metric_set(sens, yardstick::spec, accuracy, roc_auc),
    control = control_resamples(save_pred = TRUE, event_level = 'second'))

# find accuracy measures with cv
collect_metrics(log_cv_fit)
```

```
## # A tibble: 4 x 6
##   .metric  .estimator  mean     n std_err .config
##   <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy binary     0.917    10 0.00898 Preprocessor1_Model1
## 2 roc_auc  binary     0.900    10 0.0130  Preprocessor1_Model1
## 3 sens     binary     0.467    10 0.0387  Preprocessor1_Model1
## 4 spec     binary     0.976    10 0.00402 Preprocessor1_Model1
```

Random Forest

We believe that using random forest as a classification to answer our research question: Can we predict the method of execution? Would be one effective method. We reached this conclusion by evaluating the outcome variable and the nature of the research question.

Outcome variable: Method of execution → Categorical outcome → Classification → Classification tree → Random forest → Bagging (bootstrap to improve statistical learning methods such as decision trees)

Our outcome variable is the method of execution, since it is a categorical outcome we will use a classification tree to answer our research question. To limit bias we will specifically use Random forest and the bootstrap method, to improve the statistical learning method.

Since we are dealing with several predictor variables such as region, year, and race, a classification tree will be able to produce a sequence of rules that can be used to classify the data and predict the method of execution. Classification trees are easy to interpret, imitate the human decision-making process, and handle qualitative predictions without the need to create dummy variables.

```
set.seed(123)
```

```r
executions_clean <- executions_clean %>% select(-Electrocution)

# Model Specification
rf_spec <- rand_forest() %>%
  set_engine(engine = 'ranger') %>%
  set_args(mtry = NULL,
           trees = 1000,
           min_n = 2,
           probability = FALSE,
           importance = 'impurity') %>%
  set_mode('classification')

# Recipe
data_rec <- recipe(Method ~ ., data = executions_clean)

# Workflows
data_wf_mtry2 <- workflow() %>%
  add_model(rf_spec %>% set_args(mtry = 2)) %>%
  add_recipe(data_rec)

## Create workflows for different variable numbers
data_wf_mtry8 <- workflow() %>% add_model(rf_spec %>% set_args(mtry = 8)) %>% add_recipe(data_rec)

data_wf_mtry12 <- workflow() %>% add_model(rf_spec %>% set_args(mtry = 12)) %>% add_recipe(data_rec)

data_wf_mtry19 <- workflow() %>% add_model(rf_spec %>% set_args(mtry = 19)) %>% add_recipe(data_rec)
```

Implementing Random Forest When developing the model, we selected thresholds for the predictor variables to determine the method of execution. We used bagging to generate a tree that does not overfit, thereafter implemented OOB to test on and make the model is not overfitting. Bagging is resampling with replacement and random forest allowed us to create many samples which means less overfitting. By splitting on the threshold that we have set for all the predictors, we were able to predict the method of execution by the different predictor variables. The trees will split until the Gini purity index is low, or in other words, there is just one category in each leaf

```r
# Fit Models
set.seed(123)
data_fit_mtry2 <- fit(data_wf_mtry2, data = executions_clean)

set.seed(123)
data_fit_mtry8 <- fit(data_wf_mtry8, data=executions_clean)

set.seed(123)
data_fit_mtry12 <- fit(data_wf_mtry12, data=executions_clean)

set.seed(123)
data_fit_mtry19 <- fit(data_wf_mtry19, data=executions_clean)
```

```r
# Custom Function to get OOB predictions, true observed outcomes and add a user-provided model label
rf_OOB_output <- function(fit_model, model_label, truth){
    tibble(
          .pred_class = fit_model %>% extract_fit_engine() %>% pluck('predictions'), #OOB predictions
          class = truth,
```

```
          label = model_label
      )
}
```

Conclusions

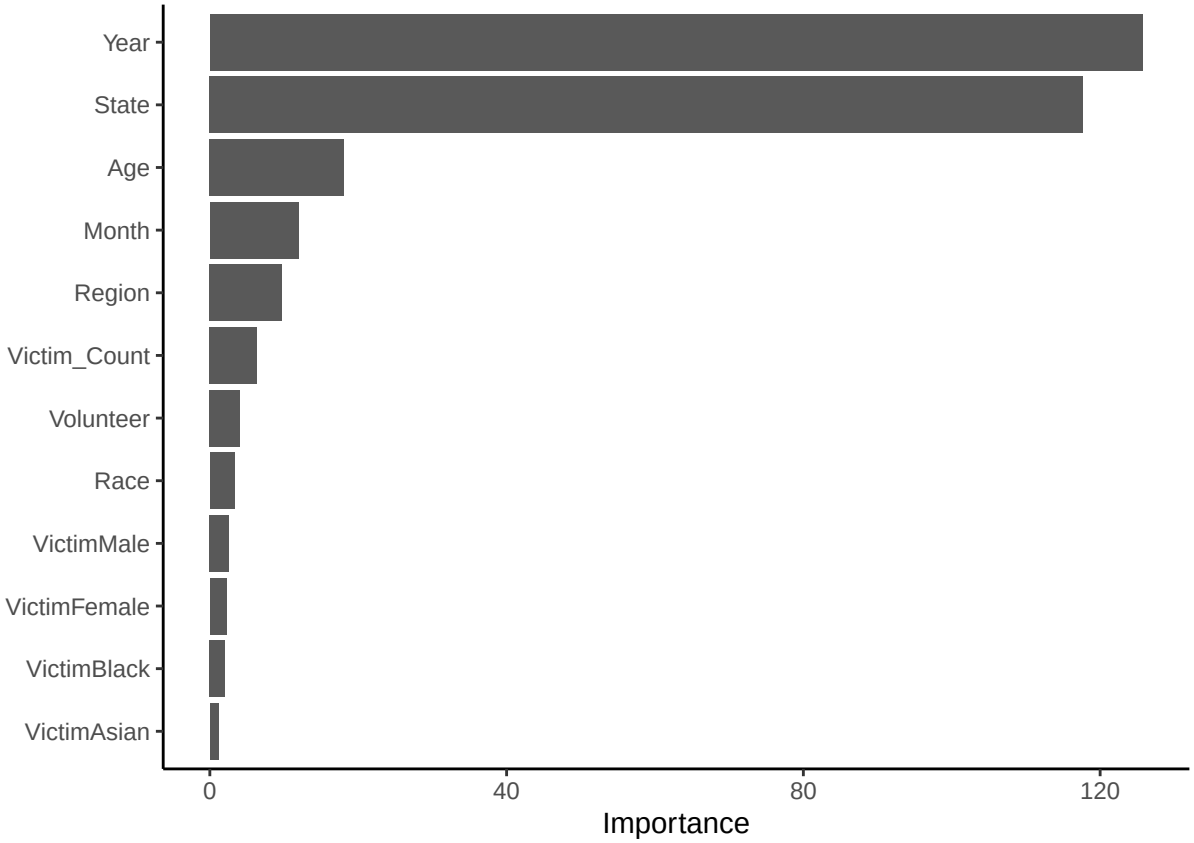
To evaluate our model we will use statistical measures such as the classification error rate and the Gini index. The classification error rate is the fraction of the training observations in that region that do not belong to the most common class. The Gini index measures the node purity. Although the classification error rate is preferable for the prediction accuracy of the final pruned tree, we measure the Gini index as well since it is more sensitive to node purity than is the classification error rate.

```
# Evaluate OOB Metrics
data_rf_OOB_output <- bind_rows(
    rf_OOB_output(data_fit_mtry2,2, executions_clean %>% pull(Method)),
    rf_OOB_output(data_fit_mtry8,8, executions_clean %>% pull(Method)),
    rf_OOB_output(data_fit_mtry12,12, executions_clean %>% pull(Method)),
    rf_OOB_output(data_fit_mtry19,19, executions_clean %>% pull(Method))
)

data_rf_OOB_output$class <-
  factor(data_rf_OOB_output$class)

output <- data_rf_OOB_output %>%
    group_by(label) %>%
    accuracy(truth = class, estimate = .pred_class)

output
```

```
## # A tibble: 4 x 4
##   label .metric  .estimator .estimate
##   <dbl> <chr>    <chr>          <dbl>
## 1     2 accuracy multiclass     0.923
## 2     8 accuracy multiclass     0.969
## 3    12 accuracy multiclass     0.970
## 4    19 accuracy multiclass     0.970
```

Overall, the out of bag error shows that the model that uses 12 variables has the greatest accuracy at 97%. Year is the most important variable for classification, which makes sense because it dictates customs and rules, and Federal and Crime type are the least important which is unexpected since we would think that punishment has to do with severity of crime.

```
#measure variable importance
model_output <-data_fit_mtry12 %>%
    extract_fit_engine()

model_output %>%
    vip(num_features = 12) + theme_classic() #based on impurity
```




```
model_output %>% vip::vi() %>% head()
```

```
## # A tibble: 6 x 2
##   Variable     Importance
##   <chr>             <dbl>
## 1 Year             126.
## 2 State            118.
## 3 Age               18.1
## 4 Month             11.9
## 5 Region             9.73
## 6 Victim_Count       6.36
```

```
model_output %>% vip::vi() %>% tail()
```

```
## # A tibble: 6 x 2
##   Variable         Importance
##   <chr>                 <dbl>
## 1 Foreign_National      0.891
## 2 Sex                   0.875
## 3 VictimLatino          0.329
## 4 Juvenile              0.211
## 5 Crime                 0
## 6 Federal               0
```

Overall, our random forest model with 8 variables was the best classification model we could come up with to predict the type of execution being performed. Out of bag error was left with over 97% accuracy and allowed the model to not overfit while still being complex.

Unsupervised Learning

Clustering

```
executions_clean <- executions %>%
  select(-Name, -County) %>%
  filter(Victim_Count < 50) %>%
  separate(Date, c("Month", "Day", "Year"), sep = "/") %>%
  mutate(VictimMale = case_when(
    str_detect(Victim_Sex, "Male") ~ 1, TRUE ~ 0)) %>% # 1 if there was male victim(s)
  mutate(VictimFemale = case_when(
    str_detect(Victim_Sex, "Female") ~ 1, TRUE ~ 0)) %>% # 1 if there was female victim(s)
  mutate(VictimWhite = case_when(
    str_detect(Victim_Race, 'White') ~ 1, TRUE ~ 0)) %>%
  mutate(VictimLatino = case_when(
    str_detect(Victim_Race, 'Latino') ~ 1, TRUE ~ 0)) %>%
  mutate(VictimBlack = case_when(
    str_detect(Victim_Race, 'Black') ~ 1, TRUE ~ 0)) %>%
  mutate(VictimAsian = case_when(
    str_detect(Victim_Race, 'Asian') ~ 1, TRUE ~ 0)) %>%
  mutate(Electrocution = case_when(
    str_detect(Method, 'Electrocution') ~ 'yes', TRUE ~ 'no')) # make a variable that shows if the pers


executions_clean <- executions_clean %>% select(-Victim_Sex, -Day, -Victim_Race)
executions_clean$Month <-
  as.numeric(as.character(executions_clean$Month)) # turn month into numeric variable
executions_clean$Year <-
  as.numeric(as.character(executions_clean$Year)) # turn year into numeric variable


head(executions)
```

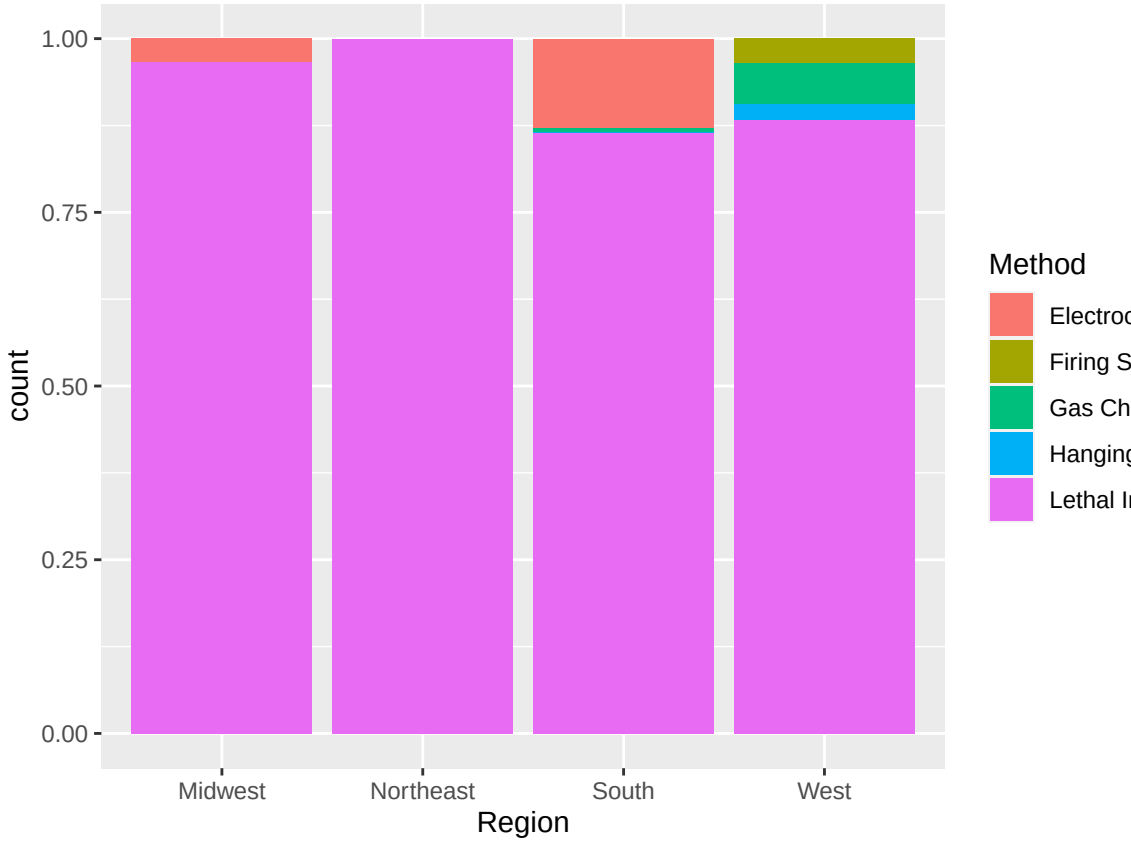
```
## # A tibble: 6 x 17
##   Date  Name    Age Sex   Race  Crime Victim_Count Victim_Sex Victim_Race County
##   <chr> <chr> <dbl> <chr> <chr> <chr>        <dbl> <chr>      <chr>       <chr>
## 1 01/1~ Gary~    36 Male  White Murd~            1 Male       White       Utah
## 2 05/2~ John~    30 Male  White Murd~            1 Male       White       Leon
## 3 10/2~ Jess~    46 Male  White Murd~            1 Male       White       Clark
## 4 03/0~ Stev~    24 Male  White Murd~            4 2 Male, 2~ White       Marion
## 5 08/1~ Fran~    38 Male  White Murd~            1 Male       White       Newpo~
## 6 12/0~ Char~    40 Male  Black Murd~            1 Male       White       Tarra~
## # ... with 7 more variables: State <chr>, Region <chr>, Method <chr>,
## #   Juvenile <chr>, Volunteer <chr>, Federal <chr>, Foreign_National <chr>
```

```
head(executions_clean)
```

```
## # A tibble: 6 x 21
##   Month  Year   Age Sex   Race  Crime  Victim_Count State Region Method Juvenile
##   <dbl> <dbl> <dbl> <chr> <chr> <chr>         <dbl> <chr> <chr>  <chr>  <chr>
## 1     1  1977    36 Male  White Murder            1 UT    West   Firin~ No
## 2     5  1979    30 Male  White Murder            1 FL    South  Elect~ No
## 3    10  1979    46 Male  White Murder            1 NV    West   Gas C~ No
## 4     3  1981    24 Male  White Murder            4 IN    Midwe~ Elect~ No
## 5     8  1982    38 Male  White Murder            1 VA    South  Elect~ No
## 6    12  1982    40 Male  Black Murder            1 TX    South  Letha~ No
## # ... with 10 more variables: Volunteer <chr>, Federal <chr>,
## #   Foreign_National <chr>, VictimMale <dbl>, VictimFemale <dbl>,
## #   VictimWhite <dbl>, VictimLatino <dbl>, VictimBlack <dbl>,
## #   VictimAsian <dbl>, Electrocution <chr>
```

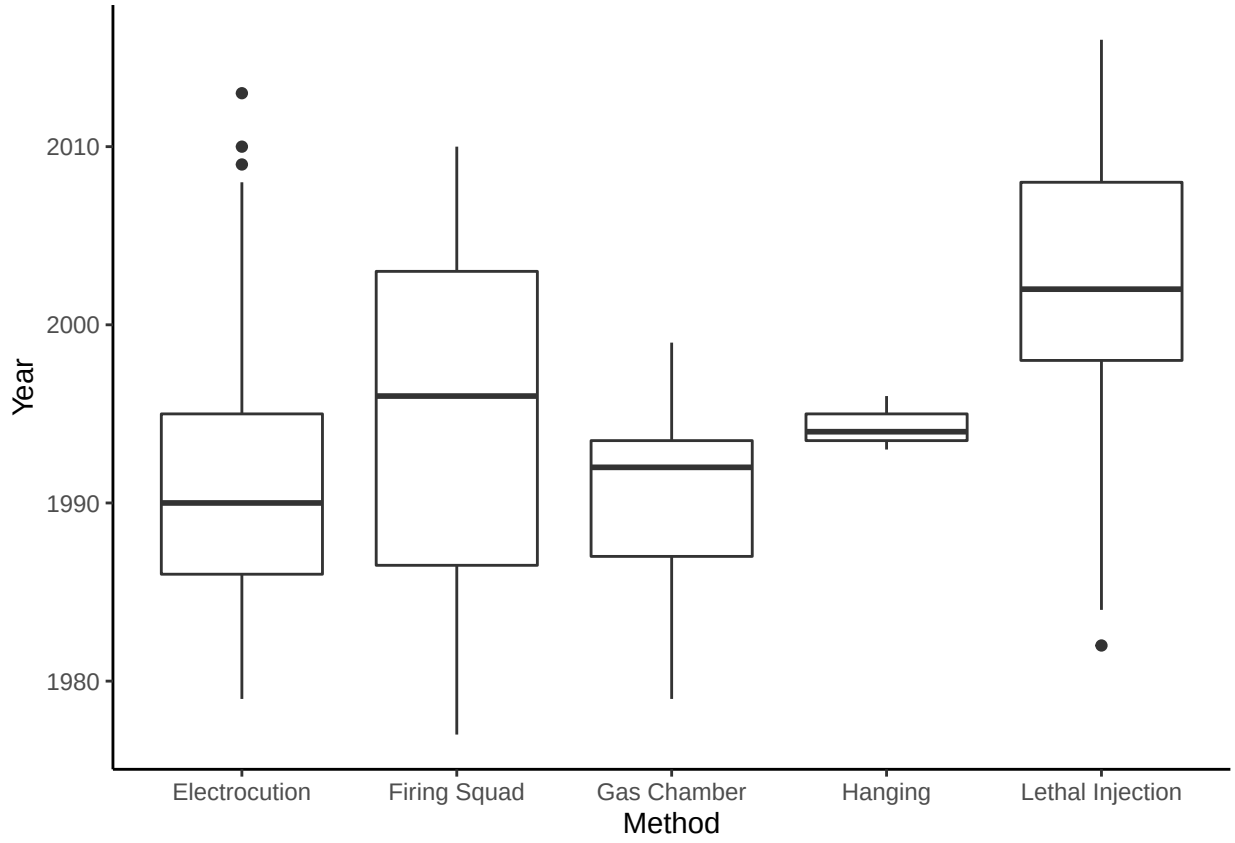
K-Means Clustering

```
# bar chart of region versus method
ggplot(executions_clean, aes(x = Region, fill = Method)) +
    geom_bar(position = 'fill')
```

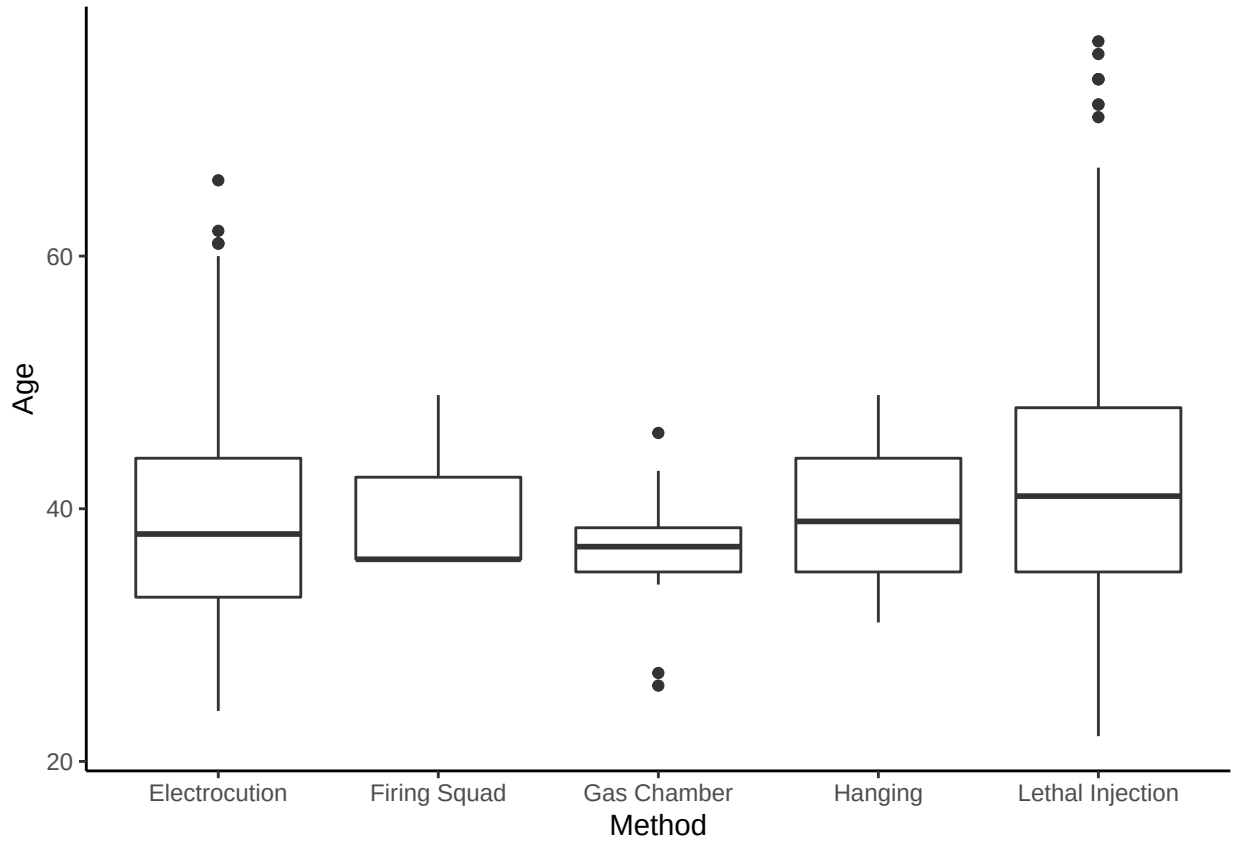


Variable Exploration

```r
# box plot of year versus method
ggplot(executions_clean, aes(x = Method, y = Year)) +
  geom_boxplot() +
  theme_classic()
```




```r
# box plot of age versus method
ggplot(executions_clean, aes(x = Method, y = Age)) +
  geom_boxplot() +
  theme_classic()
```



Age does not seem to correlate much with method, and does not improve sum of squared error in k clusters from 1-15. We created a plot of sum of squared error for k clusters with the variable age and the differences were not substantial enough to warrant further evaluation.

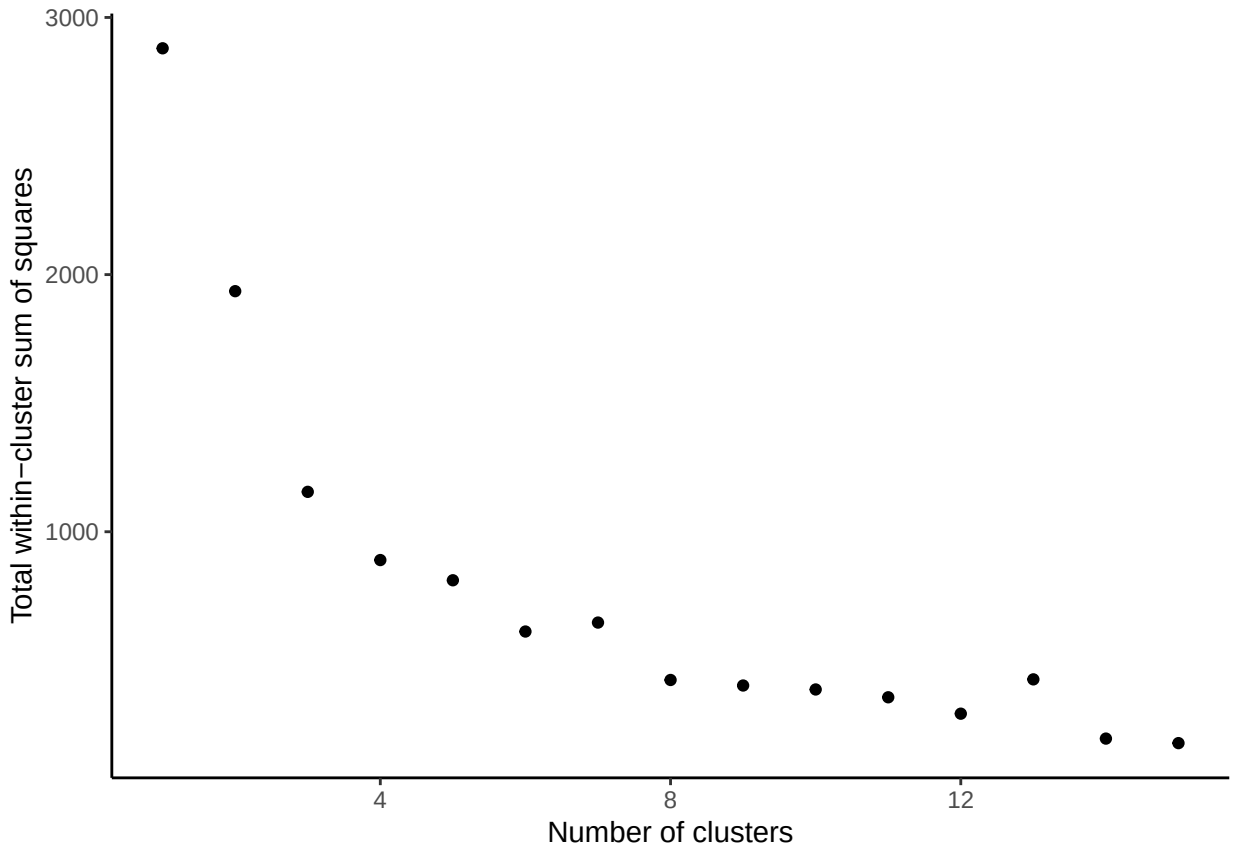
```
# Select the variables to be used in clustering
executions_sub <- executions_clean %>%
    select(Victim_Count, Year)

# Look at summary statistics of the 2 variables
summary(executions_sub)
```

```
##   Victim_Count         Year
##  Min.   : 1.000   Min.   :1977
##  1st Qu.: 1.000   1st Qu.:1997
##  Median : 1.000   Median :2001
##  Mean   : 1.422   Mean   :2001
##  3rd Qu.: 1.000   3rd Qu.:2007
##  Max.   :16.000   Max.   :2016
```

The number of victims is not very variable, whereas the year of execution fluctuates more throughout the dataset.

```
set.seed(253)
```

```
executions_cluster_ss <- function(k){
    # Perform clustering
    kclust <- kmeans(scale(executions_sub), centers = k)

    # Return the total within-cluster sum of squares
    return(kclust$tot.withinss)
}

# create table and graph number of clusters versus sum of squares
# to pick k using elbow method
tibble(
    k = 1:15,
    tot_wc_ss = purrr::map_dbl(1:15, executions_cluster_ss)
) %>%
    ggplot(aes(x = k, y = tot_wc_ss)) +
    geom_point() +
    labs(x = "Number of clusters",y = 'Total within-cluster sum of squares') +
    theme_classic()
```

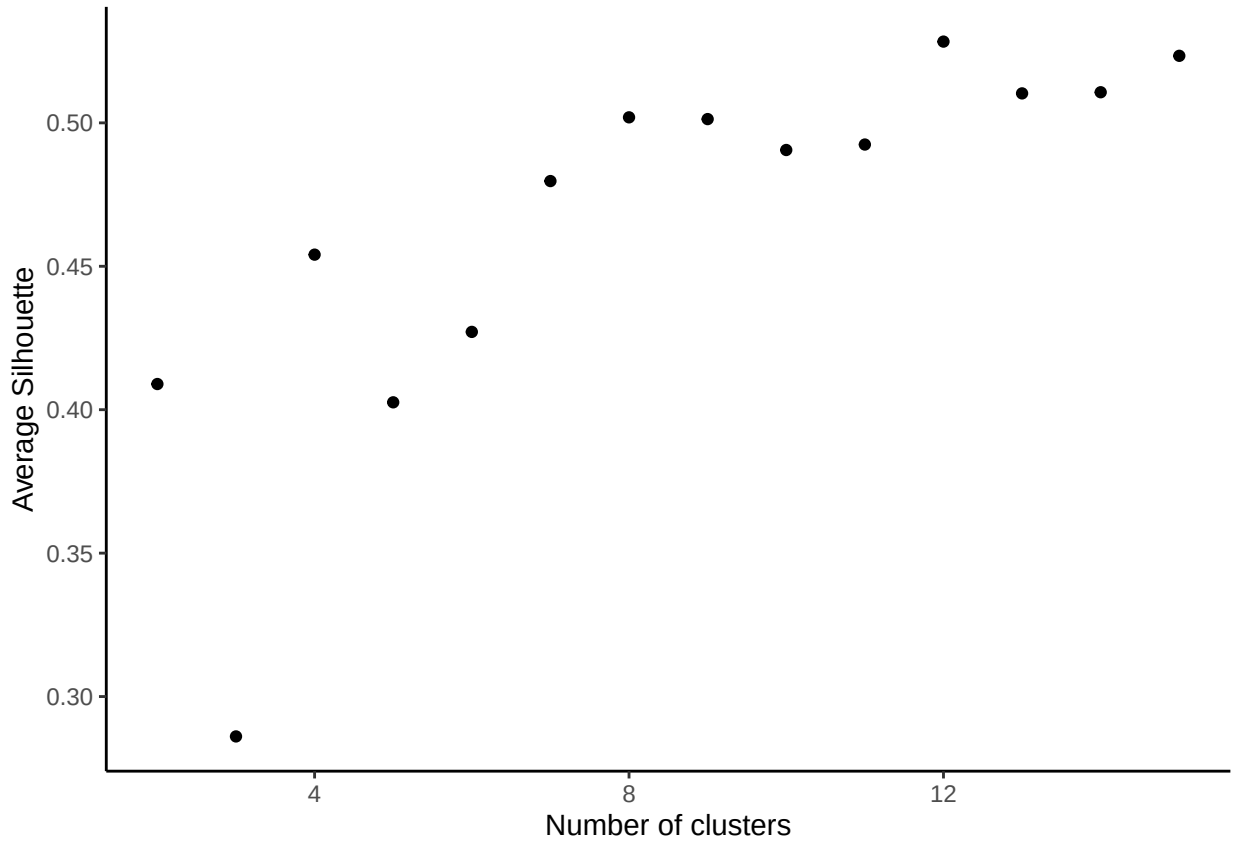


Picking *K*

According to the elbow method we would pick k = 4 or 5 (I chose 5 due to nice visual qualities it brought out when graphing later).

```
set.seed(100)

# Data-specific function to cluster and calculate total within-cluster SS
execution_cluster_silhouette <- function(k){
```

```
    # Perform clustering
    kclust <- kmeans(scale(executions_sub), centers = k)

     ss <- cluster::silhouette(kclust$cluster, dist(scale(executions_sub)))

    # Return the silhouette measures
    return(mean(ss[, 3]))
}

# Choose value that MAXIMIZES average silhouette
# table with silhouette values
sil <- tibble(
    k = 2:15,
    avg_sil = purrr::map_dbl(2:15, execution_cluster_silhouette)
)

# graph cluster versus silhouette
sil %>%
    ggplot(aes(x = k, y = avg_sil)) +
    geom_point() +
    labs(x = "Number of clusters",y = 'Average Silhouette') +
    theme_classic()
```



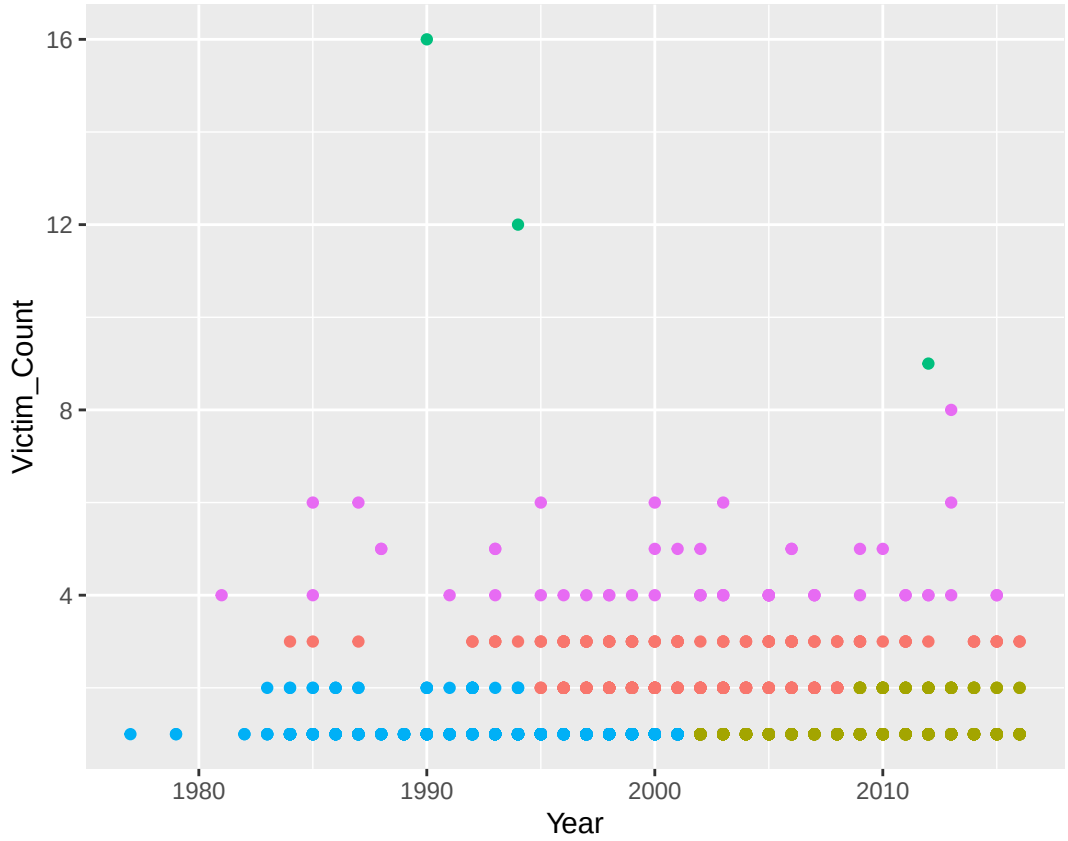

```
# view silhouette table
sil
```

```
## # A tibble: 14 x 2
##        k avg_sil
##    <int>   <dbl>
##  1     2   0.409
##  2     3   0.286
##  3     4   0.454
##  4     5   0.403
##  5     6   0.427
##  6     7   0.480
##  7     8   0.502
##  8     9   0.501
##  9    10   0.491
## 10    11   0.492
## 11    12   0.528
## 12    13   0.510
## 13    14   0.511
## 14    15   0.523
```

The value that maximized the silhouette was k = 12.

Graph K Clusters and Evaluate

```
set.seed(253)

# make the 5 clusters
kclust5 <- kmeans(scale(executions_sub), centers = 5)

# add to dataset
executions_clean <- executions_clean %>%
    mutate(kclust_5 = factor(kclust5$cluster))

# graph it
executions_clean %>%
  ggplot(aes(x = Year, y = Victim_Count, color = kclust_5)) +
  geom_point()
```



Elbow Method, K = 5

```
# print the total within-cluster sum of squares
kclust5$tot.withinss
```

```
## [1] 785.7282
```

```
# view where execution methods were in the clusters
executions_clean %>%
  count(Method,kclust_5)
```

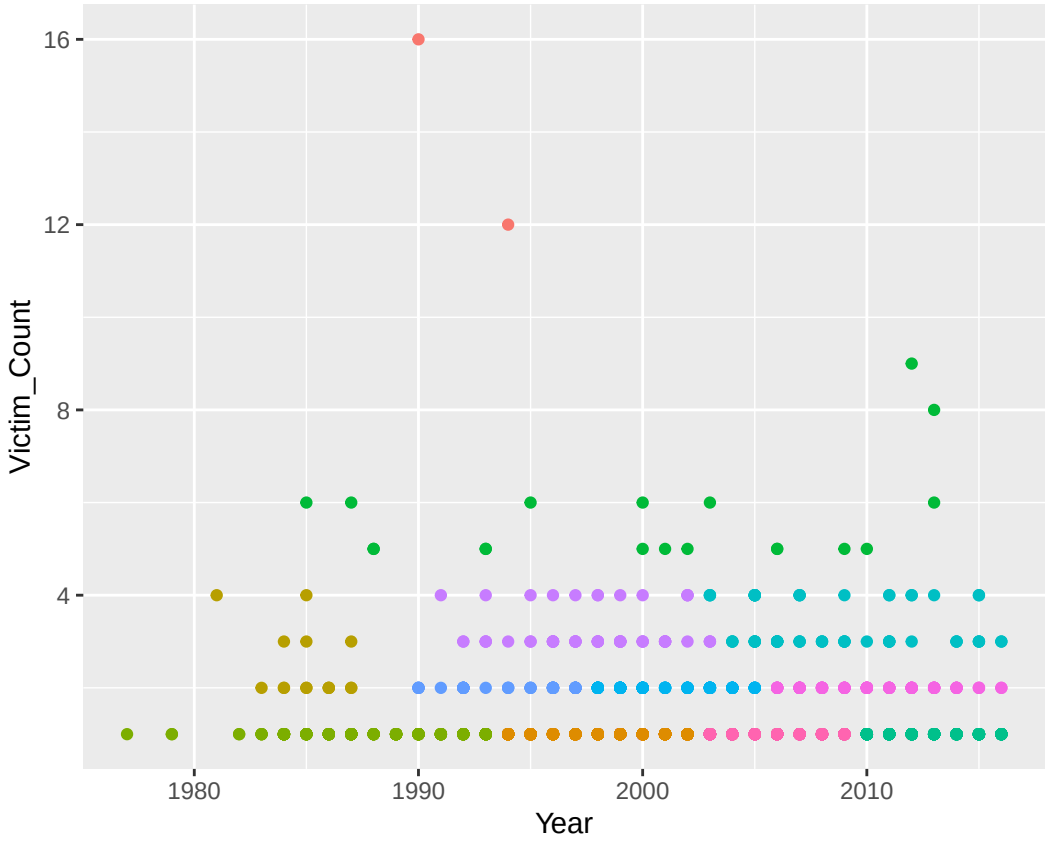
```
## # A tibble: 14 x 3
##    Method           kclust_5     n
##    <chr>            <fct>    <int>
##  1 Electrocution    1           15
##  2 Electrocution    2            5
##  3 Electrocution    4          128
##  4 Electrocution    5           10
##  5 Firing Squad     2            1
##  6 Firing Squad     4            2
##  7 Gas Chamber      4           10
##  8 Gas Chamber      5            1
##  9 Hanging          1            3
## 10 Lethal Injection 1          208
## 11 Lethal Injection 2          546
## 12 Lethal Injection 3            3
## 13 Lethal Injection 4          468
## 14 Lethal Injection 5           41
```

```r
# view how different variables average across clusters
executions_clean %>%
    group_by(kclust_5) %>%
    summarize(across(c(Year, Victim_Count), mean))
```

```
## # A tibble: 5 x 3
##   kclust_5  Year Victim_Count
##   <fct>    <dbl>        <dbl>
## 1 1        2002.         2.38
## 2 2        2008.         1.09
## 3 3        1999.        12.3
## 4 4        1995.         1.04
## 5 5        2002.         4.52
```

The k = 5 cluster model has a total sum of squared error of 785 within clusters, which is fairly low. With 5 clusters, clusters 1, 2, and 5 all represent the 2000s, with cluster 2 representing the least victim counts, and 5 representing the most. Clusters 3 and 4 represent earlier executions, but cluster 3 also represents the greatest of victim counts. These clusters map roughly to different execution types, showing us that execution method might have a lot to do with year and victim count. The biggest difference is that the clusters rely more on victim count than method of execution does.

```r
set.seed(100)

# make the 12 clusters
kclust12 <- kmeans(scale(executions_sub), centers = 12)

# add to dataset
executions_clean <- executions_clean %>%
    mutate(kclust_12 = factor(kclust12$cluster))

# graph it
executions_clean %>%
  ggplot(aes(x = Year, y = Victim_Count, color = kclust_12)) +
  geom_point()
```



Silhouette Method, K = 12

```
# get total within cluster sum of squares
kclust12$tot.withinss
```

```
## [1] 238.4374
```

```
# view where methods of executions were in clusters
executions_clean %>%
  count(Method,kclust_12)
```

```
## # A tibble: 32 x 3
##    Method        kclust_12     n
##    <chr>         <fct>     <int>
##  1 Electrocution 2            31
##  2 Electrocution 3            12
##  3 Electrocution 4            87
##  4 Electrocution 5             4
##  5 Electrocution 6             1
##  6 Electrocution 7             2
##  7 Electrocution 8             5
##  8 Electrocution 9             5
##  9 Electrocution 10            7
## 10 Electrocution 11            2
## # ... with 22 more rows
```

```r
# view how different variables average across clusters
executions_clean %>%
    group_by(kclust_12) %>%
    summarize(across(c(Year, Victim_Count), mean))
```

```
## # A tibble: 12 x 3
##    kclust_12  Year Victim_Count
##    <fct>     <dbl>        <dbl>
##  1 1         1992         14
##  2 2         1998.         1
##  3 3         1985          2.47
##  4 4         1989.         1
##  5 5         2000.         5.68
##  6 6         2013.         1
##  7 7         2009.         3.34
##  8 8         2001.         2
##  9 9         1995.         2
## 10 10        1998.         3.21
## 11 11        2010.         2
## 12 12        2006.         1
```

Although sum of squared error is quite a bit lower in the model with k = 12 clusters, these clusters are overfit and very hard to interpret quantatatively.