Full Code

2024-05-05

Data Cleaning

Data Combination

```
# # # # # # # # #
# DATA CLEANING #
# # # # # # # # #

# # # # # # # # # # # #
## Combine 12 Data sets

"C:/Users/paige/OneDrive/Documents/STAT 472/Team-Koopa/not combined csv files"
```

```
## [1] "C:/Users/paige/OneDrive/Documents/STAT 472/Team-Koopa/not combined csv files"
```

```
getwd()
```

```
## [1] "C:/Users/paige/OneDrive/Documents/STAT 472/Team-Koopa"
```

```
setwd("C:/Users/paige/OneDrive/Documents/STAT 472/Team-Koopa/not combined csv files")

data1 <- read.csv("Criminal_Offenses_On_campus.csv") |>
  mutate(unique_id = paste0(OPEID, "_", Campus.ID)) |>
  rename_with(~ paste0(.x,"_all_campus"), recycle0 = TRUE) |>
  rename(Survey.year = Survey.year_all_campus, unique_id = unique_id_all_campus)

data2 <- read.csv("Criminal_Offenses_On_campus_Student_Housing_Facilities.csv") |>
  mutate(unique_id = paste0(OPEID, "_", Campus.ID)) |>
  rename_with(~ paste0(.x,"_student_housing"), recycle0 = TRUE) |>
  rename(Survey.year = Survey.year_student_housing, unique_id = unique_id_student_housing)

data3 <- read.csv("Criminal_Offenses_Noncampus.csv") |>
  mutate(unique_id = paste0(OPEID, "_", Campus.ID)) |>
  rename_with(~ paste0(.x, "_crim_offense_noncampus"), recycle0 = TRUE) |>
  rename(Survey.year = Survey.year_crim_offense_noncampus, unique_id = unique_id_crim_offense_noncampus)

data4 <- read.csv("Criminal_Offenses_Public_property.csv") |>
   mutate(unique_id = paste0(OPEID, "_", Campus.ID)) |>
  rename_with(~ paste0(.x, "_crim_offense_public"), recycle0 = TRUE) |>
  rename(Survey.year = Survey.year_crim_offense_public, unique_id = unique_id_crim_offense_public)
```

```r
data5 <- read.csv("Arrests_On_campus.csv") |>
  mutate(unique_id = paste0(OPEID, "_", Campus.ID)) |>
  rename_with(~ paste0(.x, "_arrests_campus"), recycle0 = TRUE) |>
  rename(Survey.year = Survey.year_arrests_campus, unique_id = unique_id_arrests_campus)

data6 <- read.csv("Arrests_On_campus_Student_Housing_Facilities.csv") |>
  mutate(unique_id = paste0(OPEID, "_", Campus.ID)) |>
  rename_with(~ paste0(.x, "_arrests_stuhousing"), recycle0 = TRUE) |>
  rename(Survey.year = Survey.year_arrests_stuhousing, unique_id = unique_id_arrests_stuhousing)

data7 <- read.csv("Arrests_Noncampus.csv") |>
  mutate(unique_id = paste0(OPEID, "_", Campus.ID)) |>
  rename_with(~ paste0(.x, "_arrests_noncampus"), recycle0 = TRUE) |>
  rename(Survey.year = Survey.year_arrests_noncampus, unique_id = unique_id_arrests_noncampus)

data8 <- read.csv("Arrests_Public_Property.csv") |>
  mutate(unique_id = paste0(OPEID, "_", Campus.ID)) |>
  rename_with(~ paste0(.x, "_arrests_public"), recycle0 = TRUE) |>
  rename(Survey.year = Survey.year_arrests_public, unique_id = unique_id_arrests_public)

data9 <- read.csv("Disciplinary_Actions_On_campus.csv") |>
  mutate(unique_id = paste0(OPEID, "_", Campus.ID)) |>
  rename_with(~ paste0(.x, "_disciplinary_campus"), recycle0 = TRUE) |>
  rename(Survey.year = Survey.year_disciplinary_campus, unique_id = unique_id_disciplinary_campus)

setwd("C:/Users/paige/OneDrive/Documents/STAT 472/Team-Koopa")

data10 <- read.csv("Disciplinary_Actions_Student_Housing_Facilities.csv") |>
  mutate(unique_id = paste0(OPEID, "_", Campus.ID)) |>
  rename_with(~ paste0(.x, "_disciplinary_housing"), recycle0 = TRUE) |>
  rename(Survey.year = Survey.year_disciplinary_housing, unique_id = unique_id_disciplinary_housing)

setwd("C:/Users/paige/OneDrive/Documents/STAT 472/Team-Koopa/not combined csv files")

data11 <- read.csv("Disciplinary_Actions_Noncampus.csv") |>
  mutate(unique_id = paste0(OPEID, "_", Campus.ID)) |>
  rename_with(~ paste0(.x, "_disciplinary_noncampus"), recycle0 = TRUE) |>
  rename(Survey.year = Survey.year_disciplinary_noncampus, unique_id = unique_id_disciplinary_noncampus

data12 <- read.csv("Disciplinary_Actions_Public_Property.csv") |>
  mutate(unique_id = paste0(OPEID, "_", Campus.ID)) |>
  rename_with(~ paste0(.x, "_disciplinary_public"), recycle0 = TRUE) |>
  rename(Survey.year = Survey.year_disciplinary_public, unique_id = unique_id_disciplinary_public)

# This is our datasets being joined into one
dataset <- data1 |> left_join(data2) |>
  left_join(data3) |>
  left_join(data4) |>
  left_join(data5) |>
  left_join(data6) |>
  left_join(data7) |>
  left_join(data8) |>
  left_join(data9) |>
```

```
  left_join(data10) |>
  left_join(data11) |>
  left_join(data12)
```

```
## Joining with `by = join_by(Survey.year, unique_id)`
## Joining with `by = join_by(Survey.year, unique_id)`
## Joining with `by = join_by(Survey.year, unique_id)`
## Joining with `by = join_by(Survey.year, unique_id)`
## Joining with `by = join_by(Survey.year, unique_id)`
## Joining with `by = join_by(Survey.year, unique_id)`
## Joining with `by = join_by(Survey.year, unique_id)`
## Joining with `by = join_by(Survey.year, unique_id)`
## Joining with `by = join_by(Survey.year, unique_id)`
## Joining with `by = join_by(Survey.year, unique_id)`
## Joining with `by = join_by(Survey.year, unique_id)`
```

Removing Useless Columns

```
# # # # # # # # # # # # # #
## Removing Useless Columns

#remove NAs
dataset[is.na(dataset)] <- 0

#remove repeated columns (like unitid repeating for each xcel file)
#(3/4/24) just fixed some problems w this

cols_to_remove <- c("Unitid_student_housing", "Institution.name_student_housing", "OPEID_student_housing

## had to change this dataset name before removing the campses ##

cleaned <- dataset[, !names(dataset) %in% cols_to_remove]
```

Remove Campuses

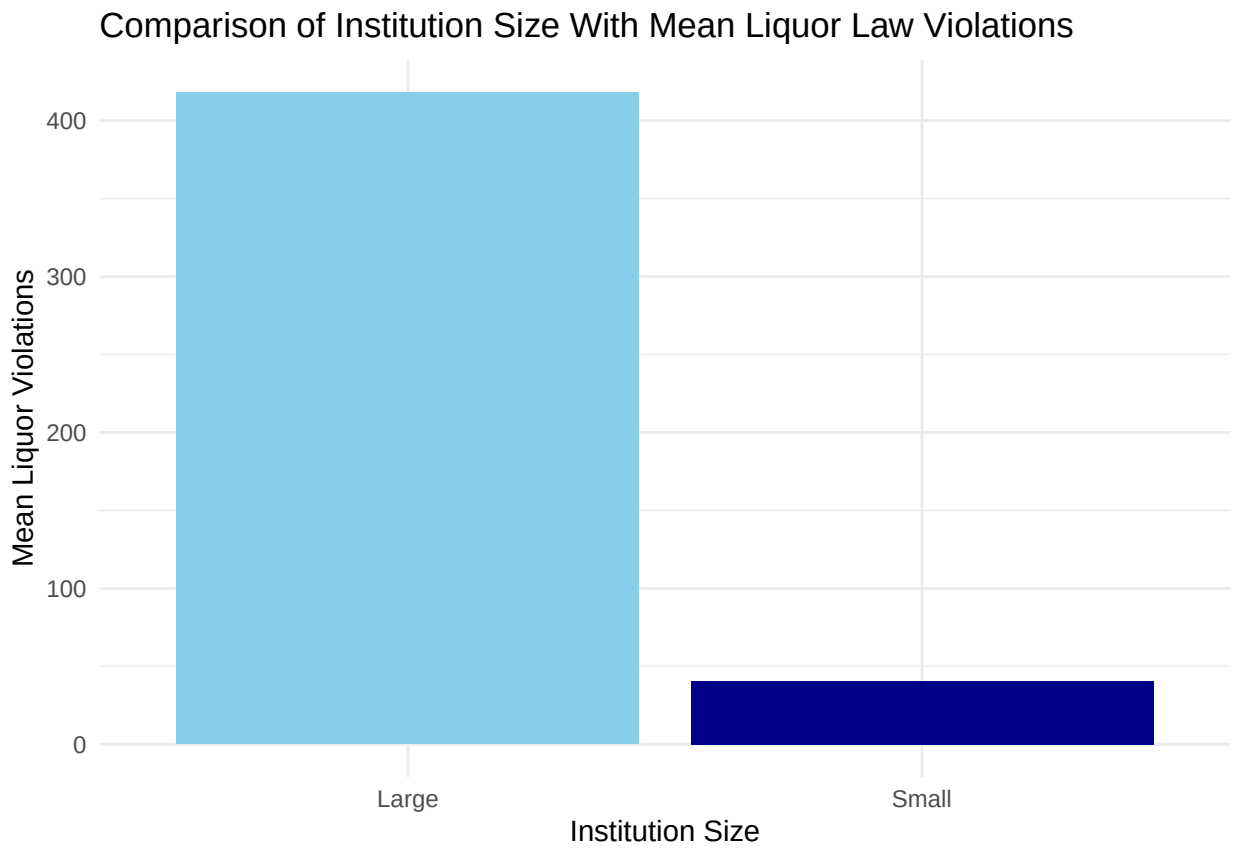
Removes campuses outside of Colorado.

```
# # # # # # # # # # # # # # # # # #
## Remove Campuses Outside of CO

# had to split into 3 vectors otherwise it's too long
to_remove1 <- c("Jacksonville", "San Diego", "Memphis", "Dunnam", "Ft. Drum", "San Luis Obispo", "Syracu

#check vector length
#length(to_remove1)

# check to see if campus name is in there
matches <- unique(grep(paste(to_remove1,collapse="|"),
                       cleaned$Campus.Name_all_campus, value=TRUE))
```

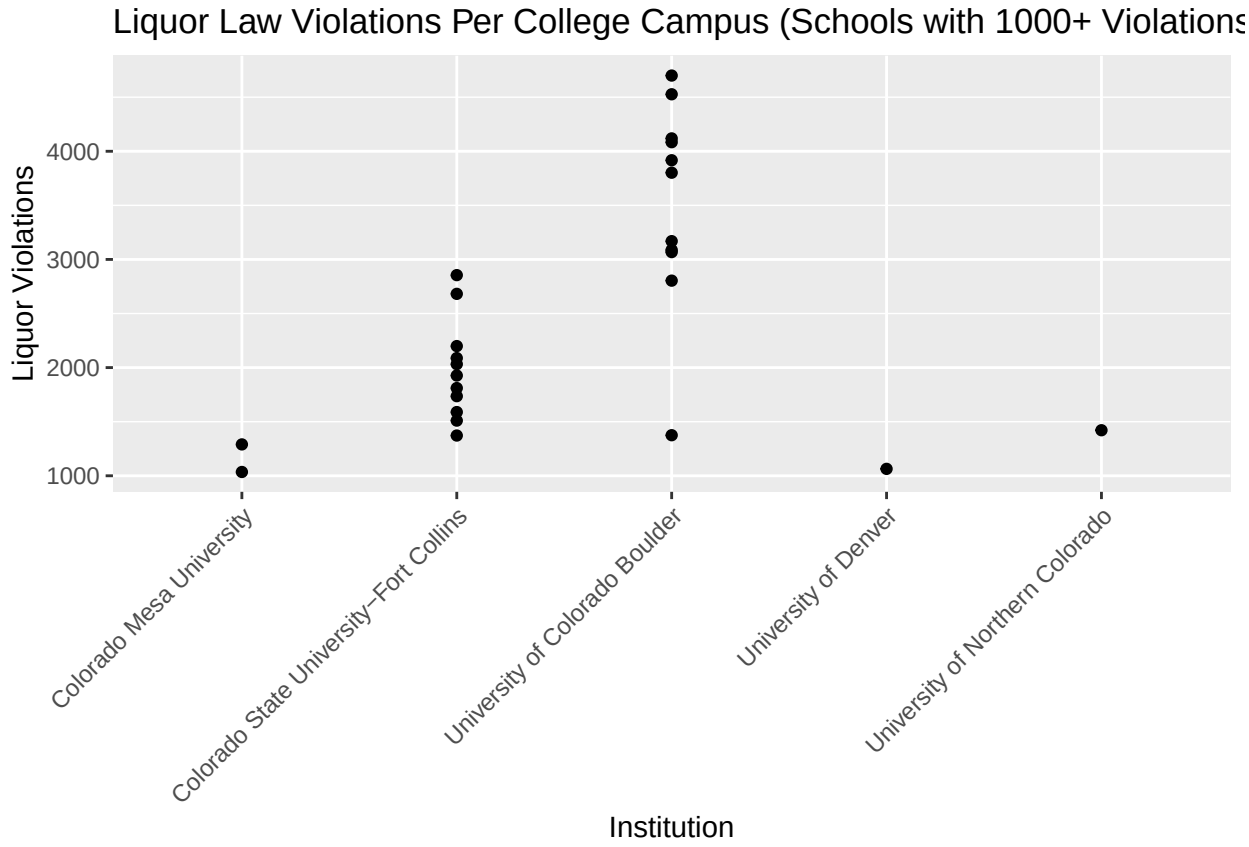
```r
# new dataset which excludes campuses from remove_1
cleaned_1 <- cleaned |> filter(!Campus.Name_all_campus %in% matches)

# same process as above
to_remove2 <- c("Albuquerque", "Wiesbaden", "Beale", "Gateway", "Ocala Metropolitan Campus", "Baton Roug

#length(to_remove2)

matches <- unique(grep(paste(to_remove2,collapse="|"),
                       cleaned_1$Campus.Name_all_campus, value=TRUE))
cleaned_2 <- cleaned_1 |> filter(!Campus.Name_all_campus %in% matches)

to_remove3 <- c("Webster University St. Louis-Main Campus", "Space Coast", "Fort Worth", "San Francisco

#length(to_remove3)

matches <- unique(grep(paste(to_remove3,collapse="|"),
                       cleaned_2$Campus.Name_all_campus, value=TRUE))

# final cleaned dataset
cleaned_data <- cleaned_2 |> filter(!Campus.Name_all_campus %in% matches)


# take a look
#head(cleaned_data)
```

Summary Statistics and EDA

```r
# # # # # # # # # # # # # ##
# SUMMARY STATISTICS & EDA #
# # # # # # # # # # # # # ##

#new column combining liquor law violations across disciplinary, arrests and location (public, stuhousi
cleaned_data$all_liquor_violations <- cleaned_data$Liquor.law.violations_arrests_campus + cleaned_data$L

numeric_data <- select(cleaned_data, where(is.numeric))

# # # # # # # # # # # # # #
## Institution Size v. TLV

institution_size <- ifelse(cleaned_data$Institution.Size_all_campus > 15000, "Large", "Small")

ggplot(cleaned_data, aes(x = institution_size, y = all_liquor_violations, fill = institution_size)) +
  geom_bar(stat = "summary", fun = "mean", position = "dodge") +
  labs(title = "Comparison of Institution Size With Mean Liquor Law Violations",
       x = "Institution Size",
       y = "Mean Liquor Violations") +
  scale_fill_manual(values = c("Large" = "skyblue", "Small" = "darkblue")) +  # Set custom fill colors
  theme_minimal() +
  theme(legend.position = "none")  # Remove legend
```




```
# # # # # # # # # # # # # # # # # # # # #
## Plot of Colleges w/ 1000+ Violations

cleaned_data |> filter(all_liquor_violations > 1000) |>
 ggplot() +
  geom_point(aes(x = Institution.name_all_campus, y = all_liquor_violations),
             color = "black") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  ggtitle("Liquor Law Violations Per College Campus (Schools with 1000+ Violations)") +
  xlab("Institution") + ylab("Liquor Violations")
```




```
# # # # # # # # # # # # # # # # # # # # # # #
## Kable Table for Means and SD of variables

# Sample data creation (assuming 'cleaned' is your data frame)
means <- round(c(mean(cleaned_data$Negligent.manslaughter_all_campus),
                 mean(cleaned_data$Sex.offenses...Forcible_all_campus),
                 mean(cleaned_data$Rape_all_campus),
                 mean(cleaned_data$Fondling_all_campus),
                 mean(cleaned_data$Sex.offenses...Non.forcible_all_campus),
                 mean(cleaned_data$Incest_all_campus),
                 mean(cleaned_data$Statutory.rape_all_campus),
                 mean(cleaned_data$Robbery_all_campus),
                 mean(cleaned_data$Burglary_all_campus),
                 mean(cleaned_data$Motor.vehicle.theft_all_campus),
                 mean(cleaned_data$Arson_all_campus)), 3)

sds <- round(c(
  sd(cleaned_data$Negligent.manslaughter_all_campus),
  sd(cleaned_data$Sex.offenses...Forcible_all_campus),
  sd(cleaned_data$Rape_all_campus),
  sd(cleaned_data$Fondling_all_campus),
  sd(cleaned_data$Sex.offenses...Non.forcible_all_campus),
  sd(cleaned_data$Incest_all_campus),
  sd(cleaned_data$Statutory.rape_all_campus),
  sd(cleaned_data$Robbery_all_campus),
  sd(cleaned_data$Burglary_all_campus),
```

```
  sd(cleaned_data$Motor.vehicle.theft_all_campus),
  sd(cleaned_data$Arson_all_campus)
), 3)

# Creating data frame
summary_df <- data.frame(
  Variable  = c("Negligent Manslaughter", "Sex Offenses (Forcible)", "Rape",
                "Fondling", "Sex Offenses (Non-forcible)", "Incest",
                "Statutory Rape", "Robbery", "Burglary", "Motor Vehicle Theft",
                "Arson"),
  Mean = means,
  StandardDeviation = sds
)

# Sorting the data frame by Mean in descending order
sorted_summary_df <- summary_df %>%
  arrange(desc(Mean), desc(StandardDeviation))

# Creating the kable
knitr::kable(sorted_summary_df, caption = "Average Values of Different Campus Offenses",
             col.names = c("Variables", "Average", "Standard Deviation"))
```

Table 1: Average Values of Different Campus Offenses

Variables	Average	Standard Deviation
Burglary	1.629	5.381
Motor Vehicle Theft	0.835	3.291
Rape	0.540	2.145
Fondling	0.354	1.476
Sex Offenses (Forcible)	0.145	1.006
Robbery	0.129	0.565
Arson	0.120	0.662
Statutory Rape	0.002	0.048
Sex Offenses (Non-forcible)	0.001	0.028
Negligent Manslaughter	0.000	0.000
Incest	0.000	0.000

```
# # # # # # # # # # # # # #
## Barplot of TLV vs. Year

year_factor <- as.factor(cleaned_data$Survey.year)

ggplot(cleaned_data, aes(x = year_factor, y = all_liquor_violations, fill = year_factor)) +
  geom_bar(stat = "identity") +
  labs(x = "Year", y = "Liquor Law Violations", fill = "Year") +
  ggtitle("Barplot of Total Liquor Violations vs. Year") +
  theme(legend.position = "none")
```

split data

```
set.seed(4242)

## split cleaned data into 25/75
smp_size <- floor(0.75 * nrow(cleaned_data))

train_split <- sample(seq_len(nrow(cleaned_data)), size = smp_size)

# create train = 75% and test = 25% set
train <- cleaned_data[train_split,] |> as_tibble() |> mutate(train = TRUE)
test <- cleaned_data[-train_split,] |> as_tibble() |> mutate(train = FALSE)


## check split to ensure nothing got screwed up

# create df of training data means and sd of each column
train_means_sd <- sapply(train[,c(7:20, 22:86)],
                         function(x) c(mean(x, na.rm = TRUE),
                                       sd(x, na.rm=TRUE)),
                         simplify = FALSE) |> bind_rows()
# transpose so table is legible
ttrain_means_sd <- t(train_means_sd)
```

```r
# create kable table
#knitr::kable(ttrain_means_sd, digits = 5, caption = "Training Data, metrics to compare to test", col.n

# create df of testing data means and sd of each column
test_means_sd <- sapply(test[,c(7:20, 22:86)],
                        function(x) c(mean(x, na.rm = TRUE),
                                      sd(x, na.rm=TRUE)),
                        simplify = FALSE) |> bind_rows()
ttest_means_sd <- t(test_means_sd)
#knitr::kable(ttest_means_sd, digits = 5, caption = "Test Data, metrics to compare to training", col.na


## kable tables for hw 5

train_means <- round(c(mean(train$Negligent.manslaughter_all_campus),
           mean(train$Sex.offenses...Forcible_all_campus),
           mean(train$Rape_all_campus),
           mean(train$Fondling_all_campus),
           mean(train$Sex.offenses...Non.forcible_all_campus),
           mean(train$Incest_all_campus),
           mean(train$Statutory.rape_all_campus),
           mean(train$Robbery_all_campus),
           mean(train$Burglary_all_campus),
           mean(train$Motor.vehicle.theft_all_campus),
           mean(train$Arson_all_campus)), 3)

train_sds <- round(c(
  sd(train$Negligent.manslaughter_all_campus),
  sd(train$Sex.offenses...Forcible_all_campus),
  sd(train$Rape_all_campus),
  sd(train$Fondling_all_campus),
  sd(train$Sex.offenses...Non.forcible_all_campus),
  sd(train$Incest_all_campus),
  sd(train$Statutory.rape_all_campus),
  sd(train$Robbery_all_campus),
  sd(train$Burglary_all_campus),
  sd(train$Motor.vehicle.theft_all_campus),
  sd(train$Arson_all_campus)
), 3)

train_pres <- data.frame(
  Variable  = c("Negligent Manslaughter", "Sex Offenses (Forcible)", "Rape",
                "Fondling", "Sex Offenses (Non-forcible)", "Incest",
                "Statutory Rape", "Robbery", "Burglary", "Motor Vehicle Theft",
                "Arson"),
  Mean = train_means,
  StandardDeviation = train_sds
)

knitr::kable(train_pres, caption = "Training Data", col.names = c("Variable", "Mean", "SD"))
```

Table 2: Training Data

Variable	Mean	SD
Negligent Manslaughter	0.000	0.000
Sex Offenses (Forcible)	0.131	0.988
Rape	0.514	2.041
Fondling	0.332	1.362
Sex Offenses (Non-forcible)	0.000	0.000
Incest	0.000	0.000
Statutory Rape	0.002	0.046
Robbery	0.137	0.581
Burglary	1.555	5.217
Motor Vehicle Theft	0.826	3.259
Arson	0.103	0.639

```
test_means <- round(c(mean(test$Negligent.manslaughter_all_campus),
           mean(test$Sex.offenses...Forcible_all_campus),
           mean(test$Rape_all_campus),
           mean(test$Fondling_all_campus),
           mean(test$Sex.offenses...Non.forcible_all_campus),
           mean(test$Incest_all_campus),
           mean(test$Statutory.rape_all_campus),
           mean(test$Robbery_all_campus),
           mean(test$Burglary_all_campus),
           mean(test$Motor.vehicle.theft_all_campus),
           mean(test$Arson_all_campus)), 3)

test_sds <- round(c(
  sd(test$Negligent.manslaughter_all_campus),
  sd(test$Sex.offenses...Forcible_all_campus),
  sd(test$Rape_all_campus),
  sd(test$Fondling_all_campus),
  sd(test$Sex.offenses...Non.forcible_all_campus),
  sd(test$Incest_all_campus),
  sd(test$Statutory.rape_all_campus),
  sd(test$Robbery_all_campus),
  sd(test$Burglary_all_campus),
  sd(test$Motor.vehicle.theft_all_campus),
  sd(test$Arson_all_campus)
), 3)

test_pres <- data.frame(
  Variable  = c("Negligent Manslaughter", "Sex Offenses (Forcible)", "Rape",
                "Fondling", "Sex Offenses (Non-forcible)", "Incest",
                "Statutory Rape", "Robbery", "Burglary", "Motor Vehicle Theft",
                "Arson"),
  Mean = test_means,
  StandardDeviation = test_sds
)

knitr::kable(test_pres, caption = "Test Data", col.names = c("Variable", "Mean", "SD"))
```

Table 3: Test Data

Variable	Mean	SD
Negligent Manslaughter	0.000	0.000
Sex Offenses (Forcible)	0.188	1.058
Rape	0.619	2.431
Fondling	0.422	1.774
Sex Offenses (Non-forcible)	0.003	0.056
Incest	0.000	0.000
Statutory Rape	0.003	0.056
Robbery	0.106	0.514
Burglary	1.850	5.850
Motor Vehicle Theft	0.863	3.390
Arson	0.169	0.728

XGBoost

```
# # # # # #
# XGBOOST #
# # # # # #

# # # # # # # # # # # # # # # # # #
## Find Predictors Relevant to Data

# Ensure train is a data frame
if (!is.data.frame(train)) {
  stop("train must be a data frame.")
}

# Exclude specified columns
excluded_columns <- c("Survey.year", "Unitid_all_campus",
                      "OPEID_all_campus", "Campus.ID_all_campus", "Campus.Name_all_campus",
                      "Institution.Size_all_campus")

# Select only numeric columns (excluding the excluded columns)
numeric_train <- train[, sapply(train, is.numeric) &
                       !(names(train) %in% excluded_columns)]

# Calculate the sum of each numeric column
column_sums <- colSums(numeric_train, na.rm = TRUE)

# Sort the column sums from most to least
sorted_column_sums <- sort(column_sums, decreasing = TRUE)

# Print the sorted column sums
print(sorted_column_sums)
```

```
##                                  all_liquor_violations
##                                                  71475
##                Liquor.law.violations_disciplinary_campus
```

```
##                                                 30580
##                 Liquor.law.violations_disciplinary_housing
##                                                 29262
##                    Drug.law.violations_disciplinary_campus
##                                                 13695
##                   Drug.law.violations_disciplinary_housing
##                                                 12421
##                       Liquor.law.violations_arrests_campus
##                                                  5709
##                      Drug.law.violations_arrests_campus
##                                                  4351
##                   Liquor.law.violations_arrests_stuhousing
##                                                  3620
##                     Drug.law.violations_arrests_stuhousing
##                                                  2462
##                                    Burglary_all_campus
##                                                  1491
##                      Drug.law.violations_arrests_public
##                                                  1162
##                     Liquor.law.violations_arrests_public
##                                                  1061
##              Liquor.law.violations_disciplinary_noncampus
##                                                   794
##                         Motor.vehicle.theft_all_campus
##                                                   792
##                                       Rape_all_campus
##                                                   493
##                                  Rape_student_housing
##                                                   405
##                               Burglary_student_housing
##                                                   402
##                                    Fondling_all_campus
##                                                   318
##                 Drug.law.violations_disciplinary_noncampus
##                                                   307
##                  Liquor.law.violations_disciplinary_public
##                                                   255
##                        Aggravated.assault_all_campus
##                                                   246
##                          Robbery_crim_offense_public
##                                                   197
##                     Aggravated.assault_crim_offense_public
##                                                   197
##                  Illegal.weapons.possession_arrests_campus
##                                                   197
##                    Motor.vehicle.theft_crim_offense_public
##                                                   194
##                    Liquor.law.violations_arrests_noncampus
##                                                   194
##                                  Fondling_student_housing
##                                                   168
##                  Illegal.weapons.possession_arrests_public
##                                                   153
##                                     Robbery_all_campus
```

```
##                                                 131
##                     Sex.offenses...Forcible_all_campus
##                                                 126
##          Illegal.weapons.possession_disciplinary_campus
##                                                 123
##                  Drug.law.violations_arrests_noncampus
##                                                 120
##                                      Arson_all_campus
##                                                  99
##                Sex.offenses...Forcible_student_housing
##                                                  97
##                         Burglary_crim_offense_noncampus
##                                                  94
##         Illegal.weapons.possession_disciplinary_housing
##                                                  90
##                     Aggravated.assault_student_housing
##                                                  50
##                Drug.law.violations_disciplinary_public
##                                                  48
##                                   Arson_student_housing
##                                                  39
##                              Rape_crim_offense_noncampus
##                                                  37
##              Motor.vehicle.theft_crim_offense_noncampus
##                                                  36
##                             Fondling_crim_offense_public
##                                                  26
##                           Fondling_crim_offense_noncampus
##                                                  25
##             Sex.offenses...Forcible_crim_offense_public
##                                                  24
##           Sex.offenses...Forcible_crim_offense_noncampus
##                                                  20
##                Aggravated.assault_crim_offense_noncampus
##                                                  20
##            Illegal.weapons.possession_arrests_stuhousing
##                                                  20
##                                Rape_crim_offense_public
##                                                  11
##                            Robbery_crim_offense_noncampus
##                                                   9
##              Illegal.weapons.possession_arrests_noncampus
##                                                   8
##                               Arson_crim_offense_noncampus
##                                                   6
##                                  Robbery_student_housing
##                                                   5
##                                Arson_crim_offense_public
##                                                   5
##              Murder.Non.negligent.manslaughter_all_campus
##                                                   3
##                        Motor.vehicle.theft_student_housing
##                                                   3
##         Illegal.weapons.possession_disciplinary_noncampus
```

```
##                                                     3
##                                Statutory.rape_all_campus
##                                                     2
##    Murder.Non.negligent.manslaughter_crim_offense_public
##                                                     1
##          Sex.offenses...Non.forcible_crim_offense_public
##                                                     1
##                          Negligent.manslaughter_all_campus
##                                                     0
##                   Sex.offenses...Non.forcible_all_campus
##                                                     0
##                                          Incest_all_campus
##                                                     0
##        Murder.Non.negligent.manslaughter_student_housing
##                                                     0
##                   Negligent.manslaughter_student_housing
##                                                     0
##              Sex.offenses...Non.forcible_student_housing
##                                                     0
##                                    Incest_student_housing
##                                                     0
##                            Statutory.rape_student_housing
##                                                     0
## Murder.Non.negligent.manslaughter_crim_offense_noncampus
##                                                     0
##            Negligent.manslaughter_crim_offense_noncampus
##                                                     0
##       Sex.offenses...Non.forcible_crim_offense_noncampus
##                                                     0
##                              Incest_crim_offense_noncampus
##                                                     0
##                     Statutory.rape_crim_offense_noncampus
##                                                     0
##               Negligent.manslaughter_crim_offense_public
##                                                     0
##                                Incest_crim_offense_public
##                                                     0
##                         Statutory.rape_crim_offense_public
##                                                     0
##                                Burglary_crim_offense_public
##                                                     0
##           Illegal.weapons.possession_disciplinary_public
##                                                     0
```

```
# # # # # # # # # # #
## Correlation Tests

# Check the data types of the columns
class(train$Arson_crim_offense_noncampus)
```

```
## [1] "numeric"
```

```r
class(train$all_liquor_violations)
```

```
## [1] "numeric"
```

```r
# If any of the columns are not numeric, convert them to numeric
train$Arson_crim_offense_noncampus <- as.numeric(train$Arson_crim_offense_noncampus)
train$all_liquor_violations <- as.numeric(train$all_liquor_violations)

# Perform Pearson's correlation test
correlation_result <- cor.test(train$Arson_crim_offense_noncampus, train$all_liquor_violations)

# Format the correlation test result
formatted_result <- sprintf("Correlation test result:
  - t-value = %.3f
  - degrees of freedom = %d
  - p-value = %s
  - correlation estimate = %.3f
  - 95 percent confidence interval: [%.3f, %.3f]",
  correlation_result$statistic,
  correlation_result$parameter,
  format(correlation_result$p.value),
  correlation_result$estimate,
  correlation_result$conf.int[1],
  correlation_result$conf.int[2]
)

# Print the formatted result
cat(formatted_result)
```

```
## Correlation test result:
##   - t-value = 11.064
##   - degrees of freedom = 957
##   - p-value = 7.393016e-27
##   - correlation estimate = 0.337
##   - 95 percent confidence interval: [0.279, 0.392]
```

```r
# # # # # # # # # # #
## Developing Model

# Using variables with a positive linear relationship with "all_liquor_violations"
# Define the selected variables for modeling
selected_vars <- c(
  "Liquor.law.violations_disciplinary_campus",
  "Liquor.law.violations_disciplinary_housing",
  "Drug.law.violations_disciplinary_campus",
  "Drug.law.violations_disciplinary_housing",
  "Liquor.law.violations_arrests_campus",
  "Drug.law.violations_arrests_campus",
  "Liquor.law.violations_arrests_stuhousing",
  "Drug.law.violations_arrests_stuhousing",
  "Burglary_all_campus",
  "Liquor.law.violations_disciplinary_noncampus",
```

```
  "Drug.law.violations_arrests_public",
  "Liquor.law.violations_arrests_public",
  "Motor.vehicle.theft_all_campus",
  "Burglary_student_housing",
  "Rape_all_campus",
  "Drug.law.violations_disciplinary_noncampus",
  "Rape_student_housing",
  "Liquor.law.violations_disciplinary_public",
  "Fondling_all_campus",
  "Aggravated.assault_all_campus",
  "Liquor.law.violations_arrests_noncampus",
  "Drug.law.violations_arrests_noncampus",
  "Motor.vehicle.theft_crim_offense_public",
  "Robbery_crim_offense_public",
  "Illegal.weapons.possession_arrests_campus",
  "Aggravated.assault_crim_offense_public",
  "Fondling_student_housing",
  "Sex.offenses...Forcible_all_campus",
  "Illegal.weapons.possession_disciplinary_campus",
  "Illegal.weapons.possession_arrests_public",
  "Robbery_all_campus",
  "Sex.offenses...Forcible_student_housing",
  "Arson_all_campus",
  "Illegal.weapons.possession_disciplinary_housing",
  "Burglary_crim_offense_noncampus",
  "Drug.law.violations_disciplinary_public",
  "Aggravated.assault_student_housing",
  "Arson_student_housing",
  "Motor.vehicle.theft_crim_offense_noncampus",
  "Fondling_crim_offense_public",
  "Sex.offenses...Forcible_crim_offense_public",
  "Aggravated.assault_crim_offense_noncampus",
  "Rape_crim_offense_noncampus",
  "Fondling_crim_offense_noncampus",
  "Illegal.weapons.possession_arrests_stuhousing",
  "Sex.offenses...Forcible_crim_offense_noncampus",
  "Rape_crim_offense_public",
  "Arson_crim_offense_public",
  "Robbery_crim_offense_noncampus",
  "Illegal.weapons.possession_arrests_noncampus",
  "Arson_crim_offense_noncampus"
)

# Subset the data with selected variables
# This step extracts only the columns from the dataset that are relevant for the analysis or modeling t
# It filters out unnecessary or redundant features, focusing the analysis on the variables thought to h
# This helps simplify the dataset, improves model interpretability, and potentially enhances model perf
train_subset <- train[selected_vars]

# Extract the target variable
# This step isolates the target variable "all_liquor_violations".
# "all_liquor_violations" represents the outcome that the model learns to predict based on the input fe
# By separating the target variable, the modeling process can focus on understanding the relationships
```

```r
y_train <- train$all_liquor_violations

# Train an ensemble model using XGBoost
# This step builds a predictive model using the XGBoost algorithm.
# The model is trained on the selected variables and the target variable.
# Hyperparameters like nrounds and verbose are specified to control the training process.
# Higher values for nrounds allow the model to learn more complex patterns in the data but may increase
xgb_model <- xgboost(data = as.matrix(train_subset),
                     label = y_train,
                     nrounds = 100,
                     verbose = 0)
xgb_validation1 <- xgb.cv(data = as.matrix(train_subset),
                     label = y_train,
                     nrounds = 100,
                     nfold = 10,
                     verbose = 0)
# do 5 for nrounds

# # # # # # # # # # #
## Plot for N-Rounds

# Print the trained model
print("Trained XGBoost Model:")
```

```
## [1] "Trained XGBoost Model:"
```

```r
print(xgb_model)
```

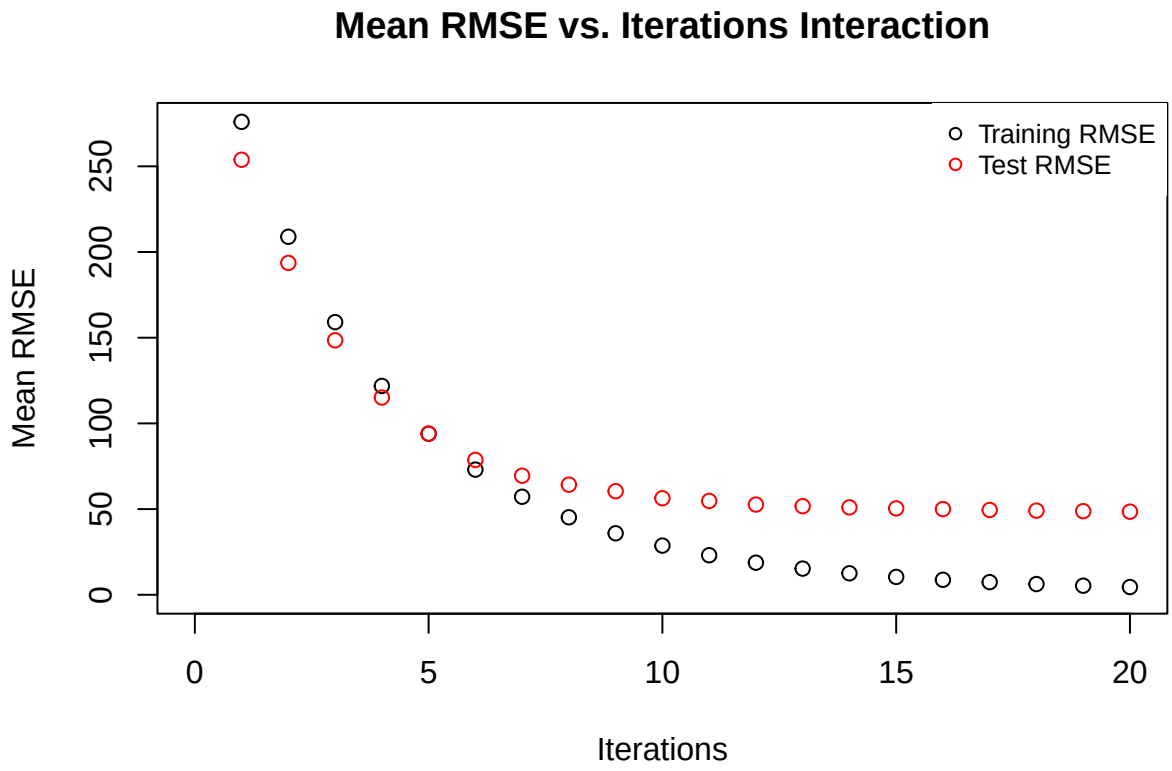
```
## ##### xgb.Booster
## raw: 223.4 Kb
## call:
##   xgb.train(params = params, data = dtrain, nrounds = nrounds,
##     watchlist = watchlist, verbose = verbose, print_every_n = print_every_n,
##     early_stopping_rounds = early_stopping_rounds, maximize = maximize,
##     save_period = save_period, save_name = save_name, xgb_model = xgb_model,
##     callbacks = callbacks)
## params (as set within xgb.train):
##   validate_parameters = "TRUE"
## xgb.attributes:
##   niter
## callbacks:
##   cb.evaluation.log()
## # of features: 51
## niter: 100
## nfeatures : 51
## evaluation_log:
##      iter   train_rmse
##     <num>        <num>
##         1 274.98191504
##         2 207.44504726
## ---
##        99   0.02935275
##       100   0.02874972
```

```
print(xgb_validation1)
```

```
## ##### xgb.cv 10-folds
##    iter train_rmse_mean train_rmse_std test_rmse_mean test_rmse_std
##   <num>           <num>          <num>          <num>         <num>
##       1    275.94794898   10.264633892      253.84008     115.27869
##       2    208.90354043    7.452532388      193.65177      91.07833
##       3    159.10223192    5.041698400      148.48200      78.28406
##       4    121.84660923    3.406365147      115.10134      64.86587
##       5     93.95985964    2.382399233       94.04568      56.02576
##       6     73.04733794    1.692061795       78.70697      49.33206
##       7     57.19742281    1.167534034       69.47830      44.43299
##       8     45.21523843    0.827575334       64.25286      42.35117
##       9     35.89375766    0.749715239       60.45085      40.67346
##      10     28.69217479    0.894607357       56.33174      37.87918
##      11     23.08621065    1.087369716       54.75696      37.24455
##      12     18.74204752    1.165826105       52.64712      36.57174
##      13     15.28128787    1.181916919       51.69804      36.38156
##      14     12.55294173    1.144842748       51.00028      36.49595
##      15     10.42059433    1.053338069       50.42045      36.68847
##      16      8.76326699    0.921084192       50.01853      36.94846
##      17      7.39342805    0.801723995       49.50114      37.21997
##      18      6.25485354    0.692932939       49.11047      37.40109
##      19      5.30776176    0.593504426       48.78062      37.52704
##      20      4.51703676    0.508691768       48.48567      37.67642
##      21      3.85514084    0.432317143       48.28084      37.79554
##      22      3.29870683    0.368753197       48.09654      37.86014
##      23      2.83010012    0.315286540       47.94571      37.97824
##      24      2.44174964    0.267061209       47.84263      38.05390
##      25      2.11459546    0.228835215       47.76150      38.13325
##      26      1.84423999    0.193263683       47.68310      38.18895
##      27      1.61266357    0.157071735       47.61471      38.21615
##      28      1.41496435    0.121199239       47.55861      38.24529
##      29      1.25673849    0.101590739       47.50847      38.27177
##      30      1.12222798    0.083817348       47.46192      38.28940
##      31      1.00960456    0.067678351       47.43677      38.30885
##      32      0.91437440    0.053951780       47.40848      38.33219
##      33      0.83485787    0.049488260       47.38638      38.34860
##      34      0.76854872    0.042265362       47.37235      38.35900
##      35      0.70731595    0.044077737       47.36664      38.37365
##      36      0.65664835    0.041326298       47.34859      38.38456
##      37      0.60742761    0.039309191       47.34479      38.39243
##      38      0.56658002    0.046068638       47.33335      38.40377
##      39      0.53167153    0.041958955       47.32050      38.41416
##      40      0.49619820    0.044403890       47.31611      38.41490
##      41      0.47207745    0.046496988       47.30879      38.41908
##      42      0.44535852    0.050050832       47.30682      38.42042
##      43      0.42001858    0.053739330       47.30119      38.42526
##      44      0.39723839    0.043332473       47.29751      38.43005
##      45      0.38614549    0.043846996       47.29307      38.43379
##      46      0.36920121    0.037946352       47.29021      38.43504
##      47      0.35218186    0.034706275       47.29451      38.43999
##      48      0.33690027    0.036038369       47.29205      38.44611
```

```
##     49      0.32051886    0.035177538       47.29033      38.44504
##     50      0.30541620    0.034780059       47.28929      38.44541
##     51      0.28593134    0.029956549       47.28897      38.44612
##     52      0.27523304    0.031878986       47.28786      38.45014
##     53      0.25991892    0.030108341       47.28767      38.45104
##     54      0.24956623    0.031768828       47.28988      38.45458
##     55      0.23406287    0.031965758       47.29056      38.45389
##     56      0.22247046    0.028781781       47.28918      38.45530
##     57      0.20869941    0.024867911       47.28900      38.45432
##     58      0.19504596    0.026197008       47.29075      38.45539
##     59      0.18555795    0.020339237       47.29162      38.45622
##     60      0.17602553    0.019587697       47.29140      38.45686
##     61      0.16818242    0.022625336       47.29047      38.45674
##     62      0.16107616    0.021362056       47.29014      38.45692
##     63      0.15202889    0.022443861       47.29008      38.45670
##     64      0.14398152    0.018648141       47.28999      38.45695
##     65      0.13815594    0.019237245       47.29042      38.45636
##     66      0.12746966    0.016175860       47.29091      38.45525
##     67      0.12208845    0.015281902       47.29134      38.45518
##     68      0.11643072    0.016893430       47.29107      38.45539
##     69      0.11058817    0.017748646       47.29162      38.45582
##     70      0.10431742    0.017423335       47.29223      38.45568
##     71      0.09874565    0.017740621       47.29243      38.45542
##     72      0.09282301    0.016010075       47.29321      38.45542
##     73      0.08806957    0.014631697       47.29316      38.45546
##     74      0.08386242    0.014233273       47.29369      38.45518
##     75      0.08111541    0.014909442       47.29360      38.45524
##     76      0.07688984    0.013186967       47.29345      38.45537
##     77      0.07379993    0.013960521       47.29362      38.45555
##     78      0.07027542    0.012944042       47.29380      38.45512
##     79      0.06711498    0.011932095       47.29362      38.45532
##     80      0.06348738    0.012281408       47.29391      38.45510
##     81      0.06083336    0.012682771       47.29400      38.45529
##     82      0.05820982    0.011900449       47.29377      38.45544
##     83      0.05492763    0.011340770       47.29361      38.45552
##     84      0.05348200    0.011433197       47.29384      38.45548
##     85      0.05190101    0.011102929       47.29402      38.45535
##     86      0.04929361    0.010112417       47.29367      38.45566
##     87      0.04766541    0.010288132       47.29352      38.45548
##     88      0.04509589    0.010162813       47.29373      38.45544
##     89      0.04298432    0.009552628       47.29368      38.45551
##     90      0.04040582    0.009362434       47.29365      38.45569
##     91      0.03923782    0.009005304       47.29367      38.45590
##     92      0.03762160    0.008578225       47.29369      38.45589
##     93      0.03596051    0.008654152       47.29367      38.45581
##     94      0.03468006    0.008224613       47.29349      38.45576
##     95      0.03350131    0.008124180       47.29313      38.45574
##     96      0.03218965    0.007877456       47.29310      38.45571
##     97      0.03100250    0.007556264       47.29318      38.45563
##     98      0.02965193    0.007133480       47.29321      38.45567
##     99      0.02848906    0.006911789       47.29324      38.45561
##    100      0.02744869    0.006984165       47.29308      38.45582
##   iter train_rmse_mean train_rmse_std test_rmse_mean test_rmse_std
```

```r
eval_metrics <- xgb_validation1$evaluation_log
plot(eval_metrics$iter, eval_metrics$train_rmse_mean,
     xlim = c(0,20),
     xlab = "Iterations",
     ylab = "Mean RMSE",
     main = "Mean RMSE vs. Iterations Interaction") # train
points(eval_metrics$iter, eval_metrics$test_rmse_mean, col = "red") # test
legend("topright", legend = c("Training RMSE", "Test RMSE"),
       col = c("black", "red"),
       pch = c(1,1),
       cex = 0.8,
       bg = "white",
       box.lty = 0)
```




```r
# # # # # # # # # # #
## Model Evaluation

xgb_model <- xgboost(data = as.matrix(train_subset),
                     label = y_train,
                     nrounds = 5,
                     verbose = 0)

test_subset <- test[selected_vars]
test_predictions <- predict(xgb_model, as.matrix(test_subset))
test_rmse <- sqrt(mean((test_predictions - test$all_liquor_violations)^2))
```

```r
cat("Test RMSE:", test_rmse, "\n")
```

```
## Test RMSE: 164.725
```

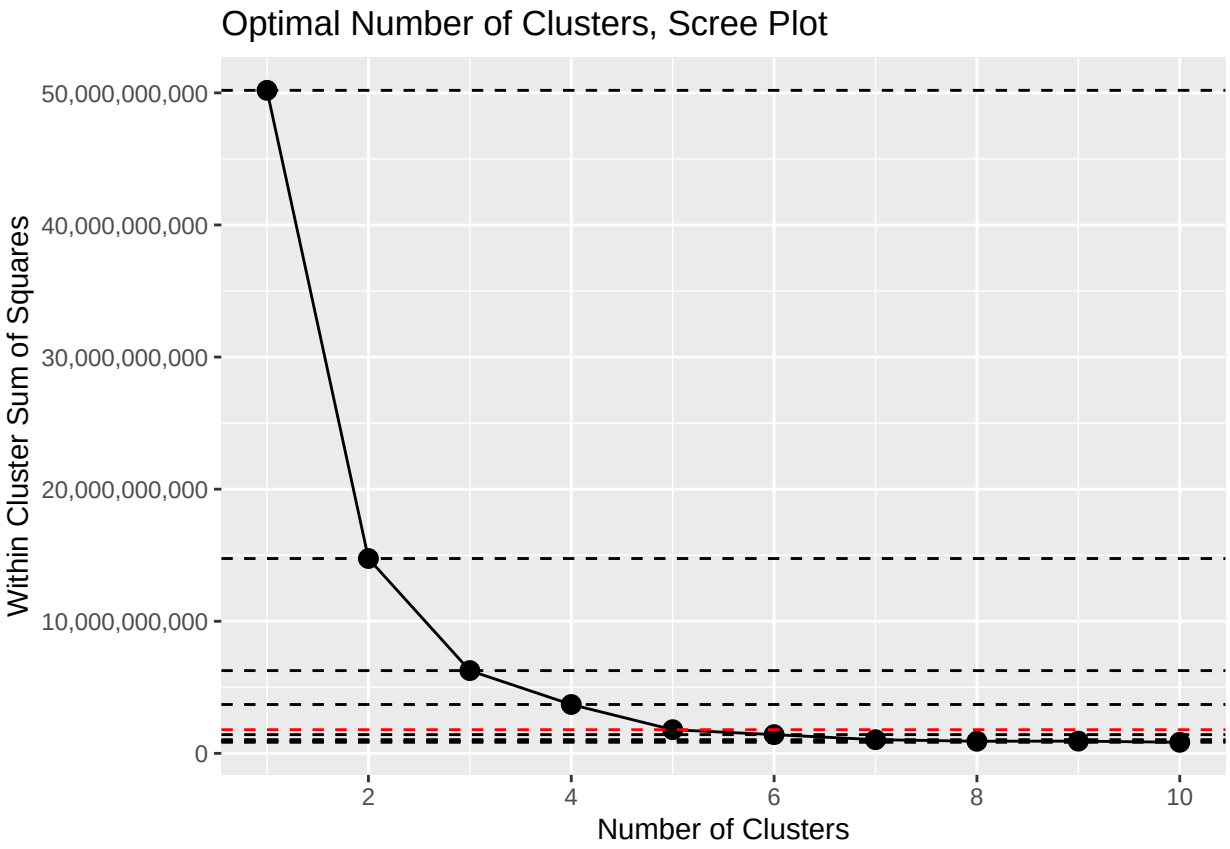
K-Means Clustering

```r
# # # # # # # # # # ##
# K-MEANS CLUSTERING #
# # # # # # # # # # ##

train_num <- train |> as_tibble() |> select(-where(is.character))

alc_2cols1 <- train_num[ , c("all_liquor_violations", "Institution.Size_all_campus")]
set.seed(421)
km.out <- kmeans(alc_2cols1, centers = 3, nstart = 20)
km.out
```

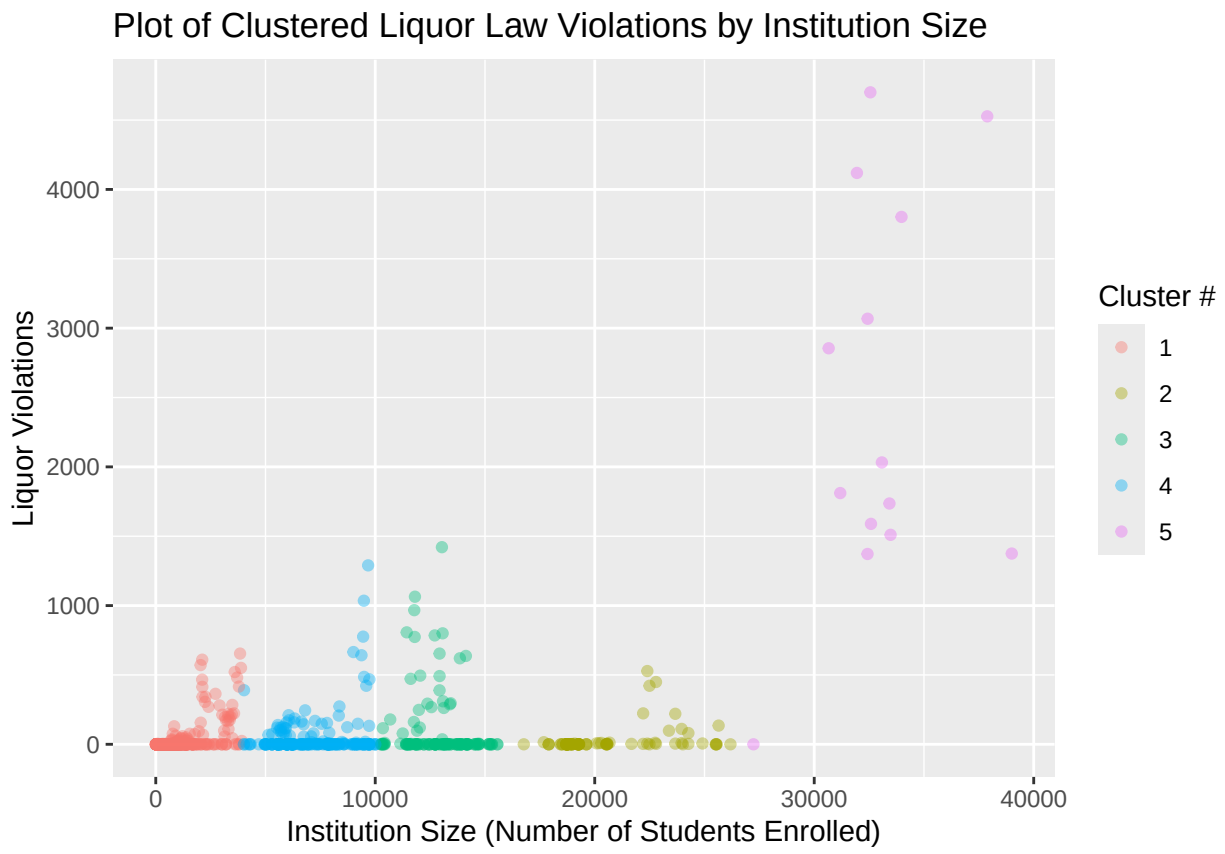
```
## K-means clustering with 3 clusters of sizes 536, 97, 326
##
## Cluster means:
##   all_liquor_violations Institution.Size_all_campus
## 1              20.76866                   976.4328
## 2             380.19588                 22326.0103
## 3              71.97546                  9736.0583
##
## Clustering vector:
##   [1] 1 1 1 3 1 1 1 1 1 1 2 2 1 1 1 1 2 3 1 3 1 3 3 3 1 1 3 1 1 3 3 1 3 3 3 3 1
##  [38] 3 1 1 1 1 3 1 1 1 1 1 3 1 3 1 1 3 3 1 1 1 3 3 1 3 1 1 1 1 3 1 1 1 1 1 1 3
##  [75] 1 1 1 1 3 2 2 1 3 3 3 2 3 1 3 3 1 1 1 3 3 1 2 1 3 1 3 1 2 3 3 1 1 2 2 3 3
## [112] 3 1 2 1 3 1 1 1 3 3 3 1 1 1 3 1 3 3 3 1 1 1 1 3 2 1 2 1 1 1 2 1 2 3 1 3 1
## [149] 3 1 1 3 1 1 3 1 1 1 3 3 3 3 1 1 1 1 1 2 1 1 3 1 1 1 3 3 2 1 2 3 1 1 1 1 3
## [186] 2 1 3 1 1 3 1 3 1 1 1 1 1 3 1 1 1 2 1 2 2 1 3 3 3 3 1 3 3 1 3 1 1 1 3 1 1
## [223] 1 1 3 2 3 3 3 1 2 1 3 1 1 2 1 1 3 1 1 1 1 1 3 3 1 1 1 1 3 3 3 1 3 1 1 1 3
## [260] 3 3 1 1 1 1 1 1 2 3 1 1 1 1 1 3 3 3 1 1 3 1 1 1 3 1 1 3 1 3 1 1 3 1 1 2 1
## [297] 1 1 1 1 1 3 1 1 1 1 3 3 1 1 1 2 1 3 3 1 1 1 3 1 2 3 3 3 1 3 2 1 1 1 1 1 1
## [334] 1 1 2 1 3 1 3 3 3 1 3 1 1 3 2 1 2 1 3 1 1 1 1 2 1 1 1 1 3 1 1 3 3 1 1 3 3
## [371] 1 1 3 1 1 3 1 3 1 1 1 2 1 3 1 3 1 1 3 1 1 2 3 3 1 1 2 1 1 2 3 1 3 3 1 3 1
## [408] 1 1 3 3 3 2 1 1 3 3 3 1 1 1 2 2 2 1 1 1 1 1 3 1 1 3 2 1 3 2 3 3 3 3 1 1 1
## [445] 1 1 1 2 1 3 3 1 1 3 3 3 1 2 3 1 1 1 1 3 1 3 3 3 2 1 3 2 1 3 3 1 2 3 3 1 1
## [482] 3 1 3 1 1 1 3 1 1 1 1 1 1 3 3 1 3 3 1 3 1 1 1 3 1 3 1 3 1 3 3 1 1 1 3 1 1
## [519] 3 1 1 3 3 1 1 1 3 1 3 1 1 3 3 1 1 3 1 1 1 3 1 3 2 3 3 3 3 1 1 3 2 1 1 1 3
## [556] 1 3 1 3 3 2 1 1 2 3 3 3 1 1 1 1 3 1 2 1 1 1 1 1 1 1 3 1 1 1 1 1 1 3 1 2 3
## [593] 1 1 1 3 2 1 1 3 1 3 2 3 1 1 3 1 3 2 1 3 1 3 3 1 3 1 1 3 3 3 1 1 3 2 1 1 1
## [630] 1 1 1 2 1 3 1 2 3 3 3 3 1 1 3 1 3 3 3 1 1 3 3 3 1 2 2 2 1 1 1 1 1 1 3 3 1
## [667] 1 1 3 1 1 3 3 1 1 1 3 3 1 1 3 3 1 3 1 1 1 1 1 1 1 3 1 2 1 3 3 2 3 1 1 3 3
## [704] 3 1 1 1 1 1 1 1 3 1 1 3 1 2 1 1 1 2 1 1 1 1 1 3 3 3 3 1 1 1 3 3 1 3 1 2 3
## [741] 3 3 3 3 3 3 1 2 3 1 3 1 1 1 1 1 1 1 3 3 3 2 1 1 1 1 3 1 1 3 3 2 1 3 3 1 3
## [778] 1 3 2 1 2 3 1 3 1 1 3 1 3 1 1 1 2 3 1 3 1 3 1 2 3 1 1 1 2 1 1 1 3 1 1 3 1
## [815] 3 1 3 3 1 1 3 3 3 3 1 1 2 1 1 1 1 1 3 1 1 3 2 2 1 1 1 3 1 2 1 2 1 2 1 2 3
## [852] 3 2 3 1 1 2 3 1 1 2 1 1 1 3 1 3 1 1 1 1 3 1 1 1 1 1 2 1 3 1 1 2 3 3 2 1 1
```

```
## [889] 3 1 1 3 1 1 3 3 1 1 1 1 1 3 3 1 1 1 1 1 1 3 1 1 3 3 3 1 3 3 3 1 1 1 3 1 3
## [926] 3 2 3 1 1 1 3 1 1 1 2 1 1 3 1 3 1 1 2 1 2 1 1 3 1 1 1 2 1 2 3 3 3 2
##
## Within cluster sum of squares by cluster:
## [1]  801514625 2520528102 2941684316
##  (between_SS / total_SS =  87.5 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```r
# # # # # # # # # # # # # # # # # # # # #
## Finding Optimal Number of Clusters

nclust <- 10
wss <- numeric(nclust)
set.seed(421)
## Looping through different number of clusters
for (i in 1:nclust) {
  km.out <- kmeans(alc_2cols1, centers = i, nstart = 20)
  wss[i] <- km.out$tot.withinss
}

## Plotting
wss_df <- tibble(clusters = 1:nclust, wss = wss)
sc_plot <- ggplot(wss_df, aes(x = clusters, y = wss, group = 1)) +
  geom_point(size = 3) +
  geom_line() +
  scale_x_continuous(breaks = c(2, 4, 6, 8, 10)) +
  scale_y_continuous(labels = scales::comma) +
  xlab("Number of Clusters") +
  ylab("Within Cluster Sum of Squares") +
  ggtitle("Optimal Number of Clusters, Scree Plot")
sc_plot +
  geom_hline(
    yintercept = wss,
    linetype = 'dashed',
    col = c(rep('black',4),'red', rep('black', 5))
  )
```




```
# # # # # # # # # # # # # # # # # # # # #
## Plotting Optimal Number of Clusters

k <- 5

set.seed(421)
km.out <- kmeans(alc_2cols1, centers = k, nstart = 20)

train_num$cluster_id <- factor(km.out$cluster)
ggplot(train_num, aes(Institution.Size_all_campus, all_liquor_violations, color = cluster_id)) +
    geom_point(alpha = 0.40) +
    xlab("Institution Size (Number of Students Enrolled)") +
    ylab("Liquor Violations") +
  ggtitle("Plot of Clustered Liquor Law Violations by Institution Size") +
  labs(color = "Cluster #")
```




```
idx <- which(km.out$cluster == 5)
train[idx,]$Institution.name_all_campus
```

```
##  [1] "University of Colorado Boulder"
##  [2] "Colorado State University-Fort Collins"
##  [3] "University of Colorado Boulder"
##  [4] "Colorado State University-Fort Collins"
##  [5] "University of Colorado Boulder"
##  [6] "Colorado Technical University-Colorado Springs"
##  [7] "Colorado State University-Fort Collins"
##  [8] "University of Colorado Boulder"
##  [9] "Colorado State University-Fort Collins"
## [10] "University of Colorado Boulder"
## [11] "University of Colorado Boulder"
## [12] "Colorado State University-Fort Collins"
## [13] "Colorado State University-Fort Collins"
## [14] "Colorado State University-Fort Collins"
```

```
idx <- km.out$cluster
idx <- which(km.out$cluster == 5)
train[idx,]
```

```
## # A tibble: 14 x 86
##    Survey.year Unitid_all_campus Institution.name_all_campus    OPEID_all_campus
##          <int>             <int> <chr>                                    <int>
```

```
##  1        2014            126614 University of Colorado Boulder           137000
##  2        2021            126818 Colorado State University-For~           135000
##  3        2012            126614 University of Colorado Boulder           137000
##  4        2019            126818 Colorado State University-For~           135000
##  5        2019            126614 University of Colorado Boulder           137000
##  6        2019            126827 Colorado Technical University~          1014800
##  7        2020            126818 Colorado State University-For~           135000
##  8        2011            126614 University of Colorado Boulder           137000
##  9        2013            126818 Colorado State University-For~           135000
## 10        2016            126614 University of Colorado Boulder           137000
## 11        2021            126614 University of Colorado Boulder           137000
## 12        2018            126818 Colorado State University-For~           135000
## 13        2017            126818 Colorado State University-For~           135000
## 14        2012            126818 Colorado State University-For~           135000
## # i 82 more variables: Campus.ID_all_campus <int>,
## #   Campus.Name_all_campus <chr>, Institution.Size_all_campus <dbl>,
## #   Murder.Non.negligent.manslaughter_all_campus <int>,
## #   Negligent.manslaughter_all_campus <int>,
## #   Sex.offenses...Forcible_all_campus <dbl>, Rape_all_campus <dbl>,
## #   Fondling_all_campus <dbl>, Sex.offenses...Non.forcible_all_campus <dbl>,
## #   Incest_all_campus <dbl>, Statutory.rape_all_campus <dbl>, ...
```

```
train[idx,]$Institution.name_all_campus
```

```
##  [1] "University of Colorado Boulder"
##  [2] "Colorado State University-Fort Collins"
##  [3] "University of Colorado Boulder"
##  [4] "Colorado State University-Fort Collins"
##  [5] "University of Colorado Boulder"
##  [6] "Colorado Technical University-Colorado Springs"
##  [7] "Colorado State University-Fort Collins"
##  [8] "University of Colorado Boulder"
##  [9] "Colorado State University-Fort Collins"
## [10] "University of Colorado Boulder"
## [11] "University of Colorado Boulder"
## [12] "Colorado State University-Fort Collins"
## [13] "Colorado State University-Fort Collins"
## [14] "Colorado State University-Fort Collins"
```

```
# # # # # # # # # # # # # # # # # #
## Clustering with 2 Predictors

test_num <- test |> as_tibble() |> select(-where(is.character))
test_num <- test_num[, !names(test_num) %in% c('Unitid_all_campus', 'OPEID_all_campus', 'Campus.ID_all_
cols_test <- train_num[ , c("all_liquor_violations", "Survey.year", "Institution.Size_all_campus")]

kmTEST <- kmeans(cols_test, centers = k, nstart = 20)

sil <- silhouette(kmTEST$cluster, dist(train_num))
```

```
## Warning in dist(train_num): NAs introduced by coercion
```

```r
data_frame <- data.frame(sil_width = sil[, "sil_width"],
                         cluster = sil[, "cluster"])
avg_sil_scores_by_cluster <- data_frame %>%
  group_by(cluster) %>%
  summarise(avg_silhouette = mean(sil_width))
print(avg_sil_scores_by_cluster)
```

```
## # A tibble: 5 x 2
##   cluster avg_silhouette
##     <dbl>          <dbl>
## 1       1          0.659
## 2       2        -0.0868
## 3       3        -0.486
## 4       4        -0.556
## 5       5         0.204
```

```r
kable(avg_sil_scores_by_cluster, format = "markdown",
      col.names =c("Cluster", "Average Silhouette Score"),
        caption = "Average Silhouette Scores by Cluster")
```

Table 4: Average Silhouette Scores by Cluster

Cluster	Average Silhouette Score
1	0.6590191
2	-0.0867991
3	-0.4855731
4	-0.5558591
5	0.2037924

```r
# # # # # #
## 3D Plot

#library(plotly)

#fig <- plot_ly(train_num, x = ~Institution.Size_all_campus, y = ~Survey.year, z = ~all_liquor_violatio
#fig <- fig %>% layout(title = "Cluster Plot of Institution Size and Survey Year",
          #            scene = list(xaxis = list(title = "Institution Size"),
           #                         yaxis = list(title = "Survey Year"),
            #                        zaxis = list(title = "Liquor Law Violations")))
#fig
```

Neural Network

```r
# # # # # # # # ##
# NEURAL NETWORK #
# # # # # # # # ##

# # # # # # # # # # # # # # # # # # # #
```

```r
## Plot Function for Clean NN Plotting

## for plot.nnet()
#install.packages("devtools")
library(devtools)
```

```
## Warning: package 'devtools' was built under R version 4.3.3
```

```
## Loading required package: usethis
```

```
## Warning: package 'usethis' was built under R version 4.3.3
```

```r
#install.packages("reshape")
library(reshape)
```

```
## Warning: package 'reshape' was built under R version 4.3.3
```

```
##
## Attaching package: 'reshape'
```

```
## The following object is masked from 'package:lubridate':
##
##     stamp
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, smiths
```

```
## The following object is masked from 'package:dplyr':
##
##     rename
```

```r
# import plot nnet function from github
plot.nnet <- function(mod.in,nid=T,all.out=T,all.in=T,bias=T,wts.only=F,rel.rsc=5,circle.cex=5,
                      node.labs=T,var.labs=T,x.lab=NULL,y.lab=NULL,line.stag=NULL,struct=NULL,cex.val=1,
                      alpha.val=1,circle.col='lightblue',pos.col='black',neg.col='grey', max.sp = F, ...)

  require(scales)

  #sanity checks
  if('mlp' %in% class(mod.in)) warning('Bias layer not applicable for rsnns object')
  if('numeric' %in% class(mod.in)){
    if(is.null(struct)) stop('Three-element vector required for struct')
    if(length(mod.in) != ((struct[1]*struct[2]+struct[2]*struct[3])+(struct[3]+struct[2])))
      stop('Incorrect length of weight matrix for given network structure')
  }
  if('train' %in% class(mod.in)){
    if('nnet' %in% class(mod.in$finalModel)){
      mod.in<-mod.in$finalModel
      warning('Using best nnet model from train output')
```

```
  }
  else stop('Only nnet method can be used with train object')
}

#gets weights for neural network, output is list
#if rescaled argument is true, weights are returned but rescaled based on abs value
nnet.vals<-function(mod.in,nid,rel.rsc,struct.out=struct){

  require(scales)
  require(reshape)

  if('numeric' %in% class(mod.in)){
    struct.out<-struct
    wts<-mod.in
  }

  #neuralnet package
  if('nn' %in% class(mod.in)){
    struct.out<-unlist(lapply(mod.in$weights[[1]],ncol))
      struct.out<-struct.out[-length(struct.out)]
      struct.out<-c(
          length(mod.in$model.list$variables),
          struct.out,
          length(mod.in$model.list$response)
          )
    wts<-unlist(mod.in$weights[[1]])
  }

  #nnet package
  if('nnet' %in% class(mod.in)){
    struct.out<-mod.in$n
    wts<-mod.in$wts
  }

  #RSNNS package
  if('mlp' %in% class(mod.in)){
    struct.out<-c(mod.in$nInputs,mod.in$archParams$size,mod.in$nOutputs)
    hid.num<-length(struct.out)-2
    wts<-mod.in$snnsObject$getCompleteWeightMatrix()

    #get all input-hidden and hidden-hidden wts
    inps<-wts[grep('Input',row.names(wts)),grep('Hidden_2',colnames(wts)),drop=F]
    inps<-melt(rbind(rep(NA,ncol(inps)),inps))$value
    uni.hids<-paste0('Hidden_',1+seq(1,hid.num))
    for(i in 1:length(uni.hids)){
      if(is.na(uni.hids[i+1])) break
      tmp<-wts[grep(uni.hids[i],rownames(wts)),grep(uni.hids[i+1],colnames(wts)),drop=F]
      inps<-c(inps,melt(rbind(rep(NA,ncol(tmp)),tmp))$value)
      }

    #get connections from last hidden to output layers
    outs<-wts[grep(paste0('Hidden_',hid.num+1),row.names(wts)),grep('Output',colnames(wts)),drop=F]
    outs<-rbind(rep(NA,ncol(outs)),outs)
```

```
    #weight vector for all
    wts<-c(inps,melt(outs)$value)
    assign('bias',F,envir=environment(nnet.vals))
    }

  if(nid) wts<-rescale(abs(wts),c(1,rel.rsc))

  #convert wts to list with appropriate names
  hid.struct<-struct.out[-c(length(struct.out))]
  row.nms<-NULL
  for(i in 1:length(hid.struct)){
    if(is.na(hid.struct[i+1])) break
    row.nms<-c(row.nms,rep(paste('hidden',i,seq(1:hid.struct[i+1])),each=1+hid.struct[i]))
  }
  row.nms<-c(
    row.nms,
    rep(paste('out',seq(1:struct.out[length(struct.out)])),each=1+struct.out[length(struct.out)-1])
    )
  out.ls<-data.frame(wts,row.nms)
  out.ls$row.nms<-factor(row.nms,levels=unique(row.nms),labels=unique(row.nms))
  out.ls<-split(out.ls$wts,f=out.ls$row.nms)

  assign('struct',struct.out,envir=environment(nnet.vals))

  out.ls

  }

wts<-nnet.vals(mod.in,nid=F)

if(wts.only) return(wts)

#circle colors for input, if desired, must be two-vector list, first vector is for input layer
if(is.list(circle.col)){
                  circle.col.inp<-circle.col[[1]]
                  circle.col<-circle.col[[2]]
                  }
else circle.col.inp<-circle.col

#initiate plotting
x.range<-c(0,100)
y.range<-c(0,100)
#these are all proportions from 0-1
if(is.null(line.stag)) line.stag<-0.011*circle.cex/2
layer.x<-seq(0.17,0.9,length=length(struct))
bias.x<-layer.x[-length(layer.x)]+diff(layer.x)/2
bias.y<-0.95
circle.cex<-circle.cex

#get variable names from mod.in object
#change to user input if supplied
if('numeric' %in% class(mod.in)){
  x.names<-paste0(rep('X',struct[1]),seq(1:struct[1]))
```

```
  y.names<-paste0(rep('Y',struct[3]),seq(1:struct[3]))
}
if('mlp' %in% class(mod.in)){
  all.names<-mod.in$snnsObject$getUnitDefinitions()
  x.names<-all.names[grep('Input',all.names$unitName),'unitName']
  y.names<-all.names[grep('Output',all.names$unitName),'unitName']
}
if('nn' %in% class(mod.in)){
  x.names<-mod.in$model.list$variables
  y.names<-mod.in$model.list$respons
}
if('xNames' %in% names(mod.in)){
  x.names<-mod.in$xNames
  y.names<-attr(terms(mod.in),'factor')
  y.names<-row.names(y.names)[!row.names(y.names) %in% x.names]
}
if(!'xNames' %in% names(mod.in) & 'nnet' %in% class(mod.in)){
  if(is.null(mod.in$call$formula)){
    x.names<-colnames(eval(mod.in$call$x))
    y.names<-colnames(eval(mod.in$call$y))
  }
  else{
    forms<-eval(mod.in$call$formula)
    x.names<-mod.in$coefnames
    facts<-attr(terms(mod.in),'factors')
    y.check<-mod.in$fitted
    if(ncol(y.check)>1) y.names<-colnames(y.check)
    else y.names<-as.character(forms)[2]
  }
}
#change variables names to user sub
if(!is.null(x.lab)){
  if(length(x.names) != length(x.lab)) stop('x.lab length not equal to number of input variables')
  else x.names<-x.lab
}
if(!is.null(y.lab)){
  if(length(y.names) != length(y.lab)) stop('y.lab length not equal to number of output variables')
  else y.names<-y.lab
}

#initiate plot
plot(x.range,y.range,type='n',axes=F,ylab='',xlab='',...)

#function for getting y locations for input, hidden, output layers
#input is integer value from 'struct'
get.ys<-function(lyr, max_space = max.sp){
  if(max_space){
      spacing <- diff(c(0*diff(y.range),0.9*diff(y.range)))/lyr
  } else {
      spacing<-diff(c(0*diff(y.range),0.9*diff(y.range)))/max(struct)
  }

      seq(0.5*(diff(y.range)+spacing*(lyr-1)),0.5*(diff(y.range)-spacing*(lyr-1)),
```

```
      length=lyr)
}

#function for plotting nodes
#'layer' specifies which layer, integer from 'struct'
#'x.loc' indicates x location for layer, integer from 'layer.x'
#'layer.name' is string indicating text to put in node
layer.points<-function(layer,x.loc,layer.name,cex=cex.val){
  x<-rep(x.loc*diff(x.range),layer)
  y<-get.ys(layer)
  points(x,y,pch=21,cex=circle.cex,col=in.col,bg=bord.col)
  if(node.labs) text(x,y,paste(layer.name,1:layer,sep=''),cex=cex.val)
  if(layer.name=='I' & var.labs) text(x-line.stag*diff(x.range),y,x.names,pos=2,cex=cex.val)
  if(layer.name=='O' & var.labs) text(x+line.stag*diff(x.range),y,y.names,pos=4,cex=cex.val)
}

#function for plotting bias points
#'bias.x' is vector of values for x locations
#'bias.y' is vector for y location
#'layer.name' is  string indicating text to put in node
bias.points<-function(bias.x,bias.y,layer.name,cex,...){
  for(val in 1:length(bias.x)){
    points(
      diff(x.range)*bias.x[val],
      bias.y*diff(y.range),
      pch=21,col=in.col,bg=bord.col,cex=circle.cex
    )
    if(node.labs)
      text(
        diff(x.range)*bias.x[val],
        bias.y*diff(y.range),
        paste(layer.name,val,sep=''),
        cex=cex.val
      )
  }
}

#function creates lines colored by direction and width as proportion of magnitude
#use 'all.in' argument if you want to plot connection lines for only a single input node
layer.lines<-function(mod.in,h.layer,layer1=1,layer2=2,out.layer=F,nid,rel.rsc,all.in,pos.col,
                      neg.col,...){

  x0<-rep(layer.x[layer1]*diff(x.range)+line.stag*diff(x.range),struct[layer1])
  x1<-rep(layer.x[layer2]*diff(x.range)-line.stag*diff(x.range),struct[layer1])

  if(out.layer==T){

    y0<-get.ys(struct[layer1])
    y1<-rep(get.ys(struct[layer2])[h.layer],struct[layer1])
    src.str<-paste('out',h.layer)

    wts<-nnet.vals(mod.in,nid=F,rel.rsc)
    wts<-wts[grep(src.str,names(wts))][[1]][-1]
```

```
    wts.rs<-nnet.vals(mod.in,nid=T,rel.rsc)
    wts.rs<-wts.rs[grep(src.str,names(wts.rs))][[1]][-1]

    cols<-rep(pos.col,struct[layer1])
    cols[wts<0]<-neg.col

    if(nid) segments(x0,y0,x1,y1,col=cols,lwd=wts.rs)
    else segments(x0,y0,x1,y1)

  }

  else{

    if(is.logical(all.in)) all.in<-h.layer
    else all.in<-which(x.names==all.in)

    y0<-rep(get.ys(struct[layer1])[all.in],struct[2])
    y1<-get.ys(struct[layer2])
    src.str<-paste('hidden',layer1)

    wts<-nnet.vals(mod.in,nid=F,rel.rsc)
    wts<-unlist(lapply(wts[grep(src.str,names(wts))],function(x) x[all.in+1]))
    wts.rs<-nnet.vals(mod.in,nid=T,rel.rsc)
    wts.rs<-unlist(lapply(wts.rs[grep(src.str,names(wts.rs))],function(x) x[all.in+1]))

    cols<-rep(pos.col,struct[layer2])
    cols[wts<0]<-neg.col

    if(nid) segments(x0,y0,x1,y1,col=cols,lwd=wts.rs)
    else segments(x0,y0,x1,y1)

  }

}

bias.lines<-function(bias.x,mod.in,nid,rel.rsc,all.out,pos.col,neg.col,...){

  if(is.logical(all.out)) all.out<-1:struct[length(struct)]
  else all.out<-which(y.names==all.out)

  for(val in 1:length(bias.x)){

    wts<-nnet.vals(mod.in,nid=F,rel.rsc)
    wts.rs<-nnet.vals(mod.in,nid=T,rel.rsc)

      if(val != length(bias.x)){
      wts<-wts[grep('out',names(wts),invert=T)]
      wts.rs<-wts.rs[grep('out',names(wts.rs),invert=T)]
          sel.val<-grep(val,substr(names(wts.rs),8,8))
          wts<-wts[sel.val]
          wts.rs<-wts.rs[sel.val]
          }
```

```
      else{
      wts<-wts[grep('out',names(wts))]
      wts.rs<-wts.rs[grep('out',names(wts.rs))]
      }

    cols<-rep(pos.col,length(wts))
    cols[unlist(lapply(wts,function(x) x[1]))<0]<-neg.col
    wts.rs<-unlist(lapply(wts.rs,function(x) x[1]))

    if(nid==F){
      wts.rs<-rep(1,struct[val+1])
      cols<-rep('black',struct[val+1])
    }

    if(val != length(bias.x)){
      segments(
        rep(diff(x.range)*bias.x[val]+diff(x.range)*line.stag,struct[val+1]),
        rep(bias.y*diff(y.range),struct[val+1]),
        rep(diff(x.range)*layer.x[val+1]-diff(x.range)*line.stag,struct[val+1]),
        get.ys(struct[val+1]),
        lwd=wts.rs,
        col=cols
      )
    }

    else{
      segments(
        rep(diff(x.range)*bias.x[val]+diff(x.range)*line.stag,struct[val+1]),
        rep(bias.y*diff(y.range),struct[val+1]),
        rep(diff(x.range)*layer.x[val+1]-diff(x.range)*line.stag,struct[val+1]),
        get.ys(struct[val+1])[all.out],
        lwd=wts.rs[all.out],
        col=cols[all.out]
      )
    }

  }
}

#use functions to plot connections between layers
#bias lines
if(bias) bias.lines(bias.x,mod.in,nid=nid,rel.rsc=rel.rsc,all.out=all.out,pos.col=alpha(pos.col,alpha
                    neg.col=alpha(neg.col,alpha.val))

#layer lines, makes use of arguments to plot all or for individual layers
#starts with input-hidden
#uses 'all.in' argument to plot connection lines for all input nodes or a single node
if(is.logical(all.in)){
  mapply(
    function(x) layer.lines(mod.in,x,layer1=1,layer2=2,nid=nid,rel.rsc=rel.rsc,
      all.in=all.in,pos.col=alpha(pos.col,alpha.val),neg.col=alpha(neg.col,alpha.val)),
    1:struct[1]
  )
```

```
  }
  else{
    node.in<-which(x.names==all.in)
    layer.lines(mod.in,node.in,layer1=1,layer2=2,nid=nid,rel.rsc=rel.rsc,all.in=all.in,
                pos.col=alpha(pos.col,alpha.val),neg.col=alpha(neg.col,alpha.val))
  }
  #connections between hidden layers
  lays<-split(c(1,rep(2:(length(struct)-1),each=2),length(struct)),
              f=rep(1:(length(struct)-1),each=2))
  lays<-lays[-c(1,(length(struct)-1))]
  for(lay in lays){
    for(node in 1:struct[lay[1]]){
      layer.lines(mod.in,node,layer1=lay[1],layer2=lay[2],nid=nid,rel.rsc=rel.rsc,all.in=T,
                  pos.col=alpha(pos.col,alpha.val),neg.col=alpha(neg.col,alpha.val))
    }
  }
  #lines for hidden-output
  #uses 'all.out' argument to plot connection lines for all output nodes or a single node
  if(is.logical(all.out))
    mapply(
      function(x) layer.lines(mod.in,x,layer1=length(struct)-1,layer2=length(struct),out.layer=T,nid=nid
                              all.in=all.in,pos.col=alpha(pos.col,alpha.val),neg.col=alpha(neg.col,alpha
      1:struct[length(struct)]
      )
  else{
    node.in<-which(y.names==all.out)
    layer.lines(mod.in,node.in,layer1=length(struct)-1,layer2=length(struct),out.layer=T,nid=nid,rel.rs
                pos.col=pos.col,neg.col=neg.col,all.out=all.out)
  }

  #use functions to plot nodes
  for(i in 1:length(struct)){
    in.col<-bord.col<-circle.col
    layer.name<-'H'
    if(i==1) { layer.name<-'I'; in.col<-bord.col<-circle.col.inp}
    if(i==length(struct)) layer.name<-'O'
    layer.points(struct[i],layer.x[i],layer.name)
    }

  if(bias) bias.points(bias.x,bias.y,'B')

}

# # # # # # # # # # # # # # #
## Fit LASSO For Predictors

set.seed(4242)

#for lasso
#install.packages("glmnet")
library(glmnet)
```

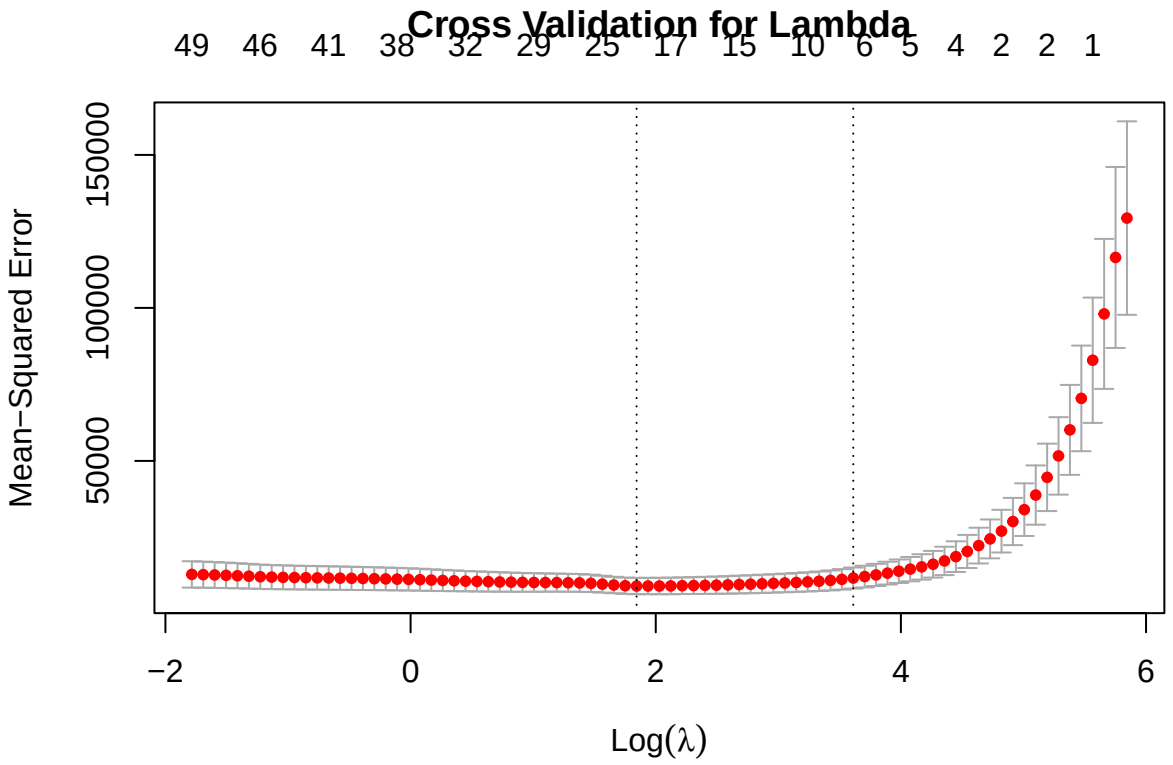
```
## Warning: package 'glmnet' was built under R version 4.3.3
```

```
## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following object is masked from 'package:reshape':
##
##     expand

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

## Loaded glmnet 4.1-8
```

```
train_num <- dplyr::select_if(train, is.numeric)

#specify y
y <- train_num$all_liquor_violations

#train$Liquor

exclude_columns <- c("Unitid_all_campus", "OPEID_all_campus",
                     "Campus.ID_all_campus", "all_liquor_violations",
                     "Liquor.law.violations_arrests_campus",
                     "Liquor.law.violations_arrests_public",
                     "Liquor.law.violations_arrests_noncampus",
                     "Liquor.law.violations_arrests_stuhousing",
                     "Liquor.law.violations_disciplinary_campus",
                     "Liquor.law.violations_disciplinary_noncampus",
                     "Liquor.law.violations_disciplinary_public",
                     "Liquor.law.violations_disciplinary_housing",
                     "new_column")

train_finalset <- train_num[, !names(train_num) %in% exclude_columns]

#specify x
x <- data.matrix(train_finalset)


# k fold cv for lambda
cv_model <- cv.glmnet(x,y,alpha = 1)
best_lambda <- cv_model$lambda.min
best_lambda
```

```
## [1] 6.320185
```

```
plot(cv_model, main = "Cross Validation for Lambda")
```




```
#find optimal lasso model
best_lasso <- glmnet(x, y, alpha = 1, lambda = best_lambda)

#coefficients from lasso model
lasso_coef <- coef(best_lasso)

lasso_coef
```

```
## 71 x 1 sparse Matrix of class "dgCMatrix"
##                                                          s0
## (Intercept)                                   -2.662625e+00
## Survey.year                                    .
## Institution.Size_all_campus                    6.232864e-04
## Murder.Non.negligent.manslaughter_all_campus   .
## Negligent.manslaughter_all_campus              .
## Sex.offenses...Forcible_all_campus             4.213089e+00
## Rape_all_campus                                .
## Fondling_all_campus                            .
## Sex.offenses...Non.forcible_all_campus         .
## Incest_all_campus                              .
## Statutory.rape_all_campus                      .
## Robbery_all_campus                             .
## Aggravated.assault_all_campus                  .
## Burglary_all_campus                            .
## Motor.vehicle.theft_all_campus                 .
## Arson_all_campus                               7.349543e+00
```

```
## Murder.Non.negligent.manslaughter_student_housing          .
## Negligent.manslaughter_student_housing                     .
## Sex.offenses...Forcible_student_housing                    .
## Rape_student_housing                                       1.319281e+01
## Fondling_student_housing                                   1.417121e+01
## Sex.offenses...Non.forcible_student_housing                .
## Incest_student_housing                                     .
## Statutory.rape_student_housing                             .
## Robbery_student_housing                                    6.750016e+01
## Aggravated.assault_student_housing                         3.563620e+01
## Burglary_student_housing                                   1.543255e+01
## Motor.vehicle.theft_student_housing                       -1.991223e+01
## Arson_student_housing                                      8.257542e+01
## Murder.Non.negligent.manslaughter_crim_offense_noncampus   .
## Negligent.manslaughter_crim_offense_noncampus              .
## Sex.offenses...Forcible_crim_offense_noncampus             .
## Rape_crim_offense_noncampus                                .
## Fondling_crim_offense_noncampus                            .
## Sex.offenses...Non.forcible_crim_offense_noncampus         .
## Incest_crim_offense_noncampus                              .
## Statutory.rape_crim_offense_noncampus                      .
## Robbery_crim_offense_noncampus                             .
## Aggravated.assault_crim_offense_noncampus                  3.253112e+01
## Burglary_crim_offense_noncampus                            .
## Motor.vehicle.theft_crim_offense_noncampus                -6.896946e+00
## Arson_crim_offense_noncampus                               8.036343e+01
## Murder.Non.negligent.manslaughter_crim_offense_public      .
## Negligent.manslaughter_crim_offense_public                 .
## Sex.offenses...Forcible_crim_offense_public                3.729253e+00
## Rape_crim_offense_public                                   .
## Fondling_crim_offense_public                               6.464286e+01
## Sex.offenses...Non.forcible_crim_offense_public            .
## Incest_crim_offense_public                                 .
## Statutory.rape_crim_offense_public                         .
## Robbery_crim_offense_public                                .
## Aggravated.assault_crim_offense_public                     .
## Burglary_crim_offense_public                               .
## Motor.vehicle.theft_crim_offense_public                    .
## Arson_crim_offense_public                                  .
## Illegal.weapons.possession_arrests_campus                  .
## Drug.law.violations_arrests_campus                         .
## Illegal.weapons.possession_arrests_stuhousing              .
## Drug.law.violations_arrests_stuhousing                     4.978611e+00
## Illegal.weapons.possession_arrests_noncampus               .
## Drug.law.violations_arrests_noncampus                      1.247533e+01
## Illegal.weapons.possession_arrests_public                  .
## Drug.law.violations_arrests_public                         .
## Illegal.weapons.possession_disciplinary_campus             .
## Drug.law.violations_disciplinary_campus                    1.109377e+00
## Illegal.weapons.possession_disciplinary_housing            .
## Drug.law.violations_disciplinary_housing                   1.473789e+00
## Illegal.weapons.possession_disciplinary_noncampus          .
## Drug.law.violations_disciplinary_noncampus                 .
## Illegal.weapons.possession_disciplinary_public             .
```

```
## Drug.law.violations_disciplinary_public                    .
```

```
#make coefficients matrix
lc_mat <- as.matrix(lasso_coef)

#make coefficients dataframe
lc_df <- as.data.frame(lc_mat)

#filter out coefficients that are 0
rows_to_keep <- apply(lc_mat, 1, function(row) any(row > 0))

lc_df_filtered <- lc_df[rows_to_keep,]

#remove intercept
lc_df_clean <- lc_df_filtered[-1]

#lc_df_clean

##lc_table_df <- data.frame(
  #Variable = c("Institution Size", "Sex Offenses (all campus)", "Arson (all campus)", "Rape (student h
  #Coefficients = lc_df_clean)

#table of lasso coefficients
#knitr::kable(lc_table_df, caption = "LASSO Coefficients", digits = 3)

# # # # # #
## Fit NNs

#install.packages("keras")
library(keras)
```

```
## Warning: package 'keras' was built under R version 4.3.3
```

```
library(tensorflow)
```

```
## Warning: package 'tensorflow' was built under R version 4.3.3
```

```
library(nnet)

#install.packages("neuralnet")

#compute object is masked from package:dplyr
library(neuralnet)
```

```
## Warning: package 'neuralnet' was built under R version 4.3.3

##
## Attaching package: 'neuralnet'

## The following object is masked from 'package:dplyr':
##
##     compute
```

```r
#get plots side by side, grid.arrange()
#install.packages("gridExtra")
library(gridExtra)

# NN test to see when model breaks
NN_1 <- neuralnet(all_liquor_violations ~ Rape_student_housing + Burglary_student_housing + Arson_studer
                  data = train, hidden = 1, linear.output=TRUE)

NN_2 <- neuralnet(all_liquor_violations ~ Rape_student_housing, hidden = 1, data = train, linear.output

NN_3 <- neuralnet(all_liquor_violations ~ Rape_student_housing + Burglary_student_housing, data = train

NN_4 <- neuralnet(all_liquor_violations ~ Rape_student_housing + Burglary_student_housing, data = train

NN_5 <- neuralnet(all_liquor_violations ~ Rape_student_housing + Burglary_student_housing + Arson_studer

NN_6 <- neuralnet(all_liquor_violations ~ Rape_student_housing + Burglary_student_housing + Drug.law.vi

plot(NN_1)

### NN plots

plot.nnet(NN_1, x.lab = c("Rape", "Burglary", "Arson", "Drug LV"), y.lab = "TLV")
```

```
## Loading required package: scales

##
## Attaching package: 'scales'

## The following object is masked from 'package:purrr':
##
##     discard

## The following object is masked from 'package:readr':
##
##     col_factor
```

```r
title("Model 1")

plot.nnet(NN_2, x.lab = c("Rape"), y.lab = "TLV")
title("Model 2")

plot.nnet(NN_3, x.lab = c("Rape", "Burglary"), y.lab = "TLV")
title("Model 3")

plot.nnet(NN_4, x.lab = c("Rape", "Burglary"), y.lab = "TLV")
title("Model 4")

plot.nnet(NN_5, x.lab = c("Rape", "Burglary", "Arson"), y.lab = "TLV")
title("Model 5")

plot.nnet(NN_6, x.lab = c("Rape", "Burglary", "Drug LV"), y.lab = "TLV")
```

```r
title("Model 6")


# # # # # # #
## NN RMSEs

library(modelr)
```

```
## Warning: package 'modelr' was built under R version 4.3.3
```

```
##
## Attaching package: 'modelr'
```

```
## The following objects are masked from 'package:Metrics':
##
##     mae, mape, mse, rmse
```

```r
## test rmse

nn_rmse <- data.frame(
  rmse_1 <- rmse(NN_1, data=test),
  rmse_2 <- rmse(NN_2, data=test),
  rmse_3 <- rmse(NN_3, data=test),
  rmse_4 <- rmse(NN_4, data=test),
  rmse_5 <- rmse(NN_5, data=test),
  rmse_6 <- rmse(NN_6, data=test)
)

new_rmse <- t(nn_rmse)

rmse_table <- data.frame(
  Variable = c("1", "2", "3", "4", "5", "6"),
  Coefficients = new_rmse)

rownames(rmse_table) <- NULL

rmse_table
```

```
##   Variable Coefficients
## 1        1     423.2550
## 2        2     436.6905
## 3        3     420.3293
## 4        4     420.3293
## 5        5     417.5463
## 6        6     423.2502
```

```r
kable(rmse_table, col.names = c("Model #", "Test RMSE"), caption = "Neural Network Model Evaluations", c
```

Table 5: Neural Network Model Evaluations

Model #	Test RMSE
1	423.255
2	436.691
3	420.329
4	420.329
5	417.546
6	423.250

Conclusion

```
# # # # # # ##
# CONCLUSION #
# # # # # # ##

final_rmse <- data.frame(
  Variable = c("XGBoost", "Neural Net"),
  Coefficients = c("164.725", "417.546"))

kable(final_rmse, col.names = c("Method", "Test RMSE"), caption = "Final Model Evaluations", digits = 3)
```

Table 6: Final Model Evaluations

Method	Test RMSE
XGBoost	164.725
Neural Net	417.546